POZNAN UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTING AND TELECOMMUNICATIONS
Institute of Radiocommunications

Doctoral thesis

CONFLICT MITIGATION IN OPEN RADIO ACCESS NETWORKS

mgr inż. Cezary Adamczyk

Supervisor
dr hab. inż. Adrian Kliks, prof. PP

POZNAŃ 2025

Acknowledgements

This dissertation is the culmination of over four years of effort dedicated to balancing professional work with academic research. This undertaking would not have been possible without the support of my closest ones: my friends and family. I thank you dearly.

I particularly wish to thank my beloved Ola—your invaluable support encouraged me to take on this challenge and helped me see it through to completion.

Special thanks are also due to my supervisor, Adrian Kliks, whose trust and guidance enabled me to achieve all my set goals.

Podziękowania

Niniejsza praca jest zwieńczeniem ponad czterech lat wysiłku polegającego na łączeniu pracy zawodowej z działalnością naukową. To poświęcenie nie byłoby możliwe, gdyby nie wsparcie moich najbliższych: przyjaciół oraz rodziny. Serdecznie wam dziękuję.

Szczególnie pragnę podziękować mojej ukochanej Oli—Twoje nieocenione wsparcie sprawiło, że zdecydowałem się na to wyzwanie i że doprowadziłem je do końca.

Specjalne podziękowania należą się również mojemu promotorowi, Adrianowi Kliksowi, którego zaufanie i porady pozwoliły mi na osiągnięcie wszystkich założonych celów.

Pracę tę dedykuję mojemu dziadkowi Januszowi.

Contents

List of Figures

List of Tables

Abstract

Instant wireless access to data has evolved into a fundamental, taken-for-granted privilege for populations in digitally advanced nations, fundamentally underpinning how modern society consumes media, gathers information, and socializes. To address this surging demand and overcome the limitations of conventional Radio Access Network (RAN) equipment – characterized by closed designs, opaque "black box" components, and vendor lock-in – Mobile Network Operators (MNOs) are increasingly adopting Open RAN (O-RAN) architectures. O-RAN introduces disaggregated, virtualized network functions with open interfaces, allowing operators to deploy base stations using combinations of hardware and software from various vendors. Central to this architecture is the introduction of robust intelligence for network optimization, implemented through Non-Real-Time (Non-RT) and Near-Real-Time (Near-RT) RAN Intelligent Controllers (RICs). These controllers host diverse applications, known as rApps and xApps, which enable operators to react to changing network conditions and adapt configurations dynamically.

The open and programmable nature of the O-RAN environment, while offering unprecedented flexibility, introduces significant operational challenges regarding conflict management. Because the O-RAN ecosystem is not limited to chosen vendors, MNOs can deploy xApps and rApps implemented by many different software providers. As these various applications operate simultaneously and modify network parameters to achieve potentially distinct optimization goals, there is a high probability that their decisions will exert conflicting influences on how the RICs steer network behavior. Such conflicts can lead to severe network performance deterioration and a waste of vital radio resources. While similar challenges exist in other complex architectures, such as operating systems managing concurrent tasks or Self-Organizing Networks (SON), the distributed nature of O-RAN logic within RICs presents a unique context that prevents the direct application of legacy solutions. Although current standards for O-RAN describe a conflict mitigation component within the Near-RT RIC and list conflict mitigation as a supporting function of the Non-RT RIC, they do not specify the detailed logic for how conflicts should be detected and resolved. Consequently, a universal methodology to detect and resolve control conflicts needs to be defined to extend the architecture currently described by the O-RAN Alliance technical documentation.

The central thesis of this dissertation is that conflict mitigation mechanisms can significantly improve the performance, stability, and reliability of O-RAN networks operating with conflicting applications. The research assumes that without effective detection and resolution measures, the benefits of the open ecosystem cannot be fully exploited. Therefore, the primary purpose of this research is to propose, implement, and evaluate a comprehensive framework for conflict detection and resolution that is universal, robust, and fully compatible with the O-RAN architecture. This involves designing a framework that fits within the foundation described in the latest standards while allowing for the resolution of all conflict types.

The methodology employed in this dissertation follows a systematic structure, beginning with a thorough literature review of O-RAN standards, technical reports, and research papers on SON function coordination to assess the state of the art. Based on identified research gaps, specifically the lack of detailed logic for conflict resolution, the study progresses to the conceptualization and design of the Conflict Mitigation Framework (CMF). This framework is designed to be a procedural, logic-agnostic solution capable of hosting various conflict detection and resolution algorithms. To assess the benefits of this framework, the research involved the design and implementation of a proprietary O-RAN network simulation environment. This simulation software models the O-RAN environment and allows for the execution of conflict scenarios under consistent conditions, enabling the fair measurement of Key Performance Indicators (KPIs).

Within the proposed framework, the research involved the implementation of multiple conflict detection and resolution algorithms to evaluate their effectiveness. Initial stages focused on implementing conventional conflict resolution methods to serve as baselines. Subsequently, the research advanced to the design and development of an Artificial Intelligence (AI) driven solution, specifically utilizing an actor-critic Artificial Neural Network (ANN) trained via Reinforcement Learning (RL). This approach allows for adaptive, data-driven conflict resolution capable of resolving complex control conflicts between xApps. The performance of these algorithms was analyzed using a novel evaluation metric proposed specifically for the comparison of conflict resolution solutions.

Beyond simulation, the dissertation validates the practical applicability of the proposed solution through hardware-based evaluation. The CMF was implemented and tested in an Open Testing and Integration Center (OTIC), a laboratory for testing O-RAN, certified by the O-RAN Alliance. This testbed utilized real O-RAN-compliant hardware and software, ensuring that the proposed solutions are effective not only in theoretical simulations but also in live network environments.

The results of this research provide a detailed description of how conflict detection and resolution can be achieved, filling the void in current O-RAN standards. The proposed frameworks represent some of the first developments in the O-RAN space dedicated to this critical issue. By providing a blueprint for ensuring network integrity in a multi-vendor environment, this work contributes significantly to the development of modern RAN technology. The conducted research can help determine how conflict mitigation will be solved in all O-RAN networks, potentially influencing how future RAN networks are optimized and ensuring that the operational flexibility of O-RAN does not come at the cost of stability.

Streszczenie

Natychmiastowy, bezprzewodowy dostęp do danych stał się oczywistym przywilejem dla mieszkańców krajów zaawansowanych cyfrowo, stanowiąc fundament sposobu, w jaki nowoczesne społeczeństwo konsumuje media, gromadzi informacje i nawiązuje kontakty społeczne. Aby sprostać temu rosnącemu popytowi i przezwyciężyć ograniczenia konwencjonalnego sprzętu sieciowego – charakteryzującego się zamkniętą konstrukcją, nieprzejrzystymi komponentami typu „black box" i uzależnieniem od jednego dostawcy (zjawisko vendor lock-in) – operatorzy sieci radiowych (ang. Mobile Network Operators, MNOs) coraz częściej wdrażają architekturę otwartej radiowej sieci dostępowej (ang. Open Radio Access Network, O-RAN). O-RAN wprowadza zdezagregowane, zwirtualizowane funkcje sieciowe z otwartymi interfejsami, umożliwiając operatorom wdrażanie stacji bazowych z wykorzystaniem kombinacji sprzętu i oprogramowania od różnych dostawców. Kluczowym elementem tej architektury jest wprowadzenie programowalnej inteligencji do optymalizacji sieci, realizowanej poprzez inteligentne sterowniki sieci (ang. Radio Intelligent Controller, RIC) działające w czasie innym niż rzeczywisty (ang. Non Real-Time RIC, Non-RT RIC) oraz w czasie zbliżonym do rzeczywistego (ang. Near Real-Time RIC, Near-RT RIC). Komponenty te pozwalają na uruchomienie różnorodnych aplikacji, nazwanych rApps i xApps, które umożliwiają operatorom reagowanie na zmieniające się warunki sieciowe i dynamiczną adaptację konfiguracji sieci.

Otwartość i programowalność sieci O-RAN, choć oferuje bezprecedensową elastyczność, wprowadza istotne wyzwania operacyjne w zakresie zarządzania konfliktami. Ponieważ ekosystem O-RAN nie ogranicza się do wybranych dostawców, MNO mogą wdrażać aplikacje xApps i rApps realizowane przez wielu różnych dostawców oprogramowania. Gdy te różnorodne aplikacje działają jednocześnie i modyfikują parametry sieci w celu osiągnięcia potencjalnie odmiennych celów optymalizacyjnych, istnieje wysokie prawdopodobieństwo, że ich decyzje będą wywierać sprzeczny wpływ na konfigurację sieci. Takie konflikty mogą prowadzić do poważnego pogorszenia wydajności sieci i marnotrawstwa zasobów radiowych. Chociaż podobne wyzwania istnieją w innych złożonych architekturach, takich jak systemy operacyjne zarządzające współbieżnymi zadaniami czy sieci Self-Organizing Networks (SON), rozproszona logika w sterownikach RIC tworzy unikalny kontekst, który uniemożliwia bezpośrednie zastosowanie rozwiązań istniejących w innych dziedzinach. Mimo że obecne standardy O-RAN opisują komponent rozwiązywania konfliktów (ang. Conflict Mitigation, ConMit) w Near-RT RIC i wymieniają rozwiązywanie konfliktów jako funkcję wspierającą w Non-RT RIC, nie precyzują one szczegółowych mechanizmów wykrywania i rozwiązywania konfliktów. W konsekwencji konieczne jest zdefiniowanie uniwersalnego sposobu wykrywania i rozwiązywania konfliktów xApp i rApp, który rozszerzałby architekturę opisaną obecnie w dokumentacji technicznej O-RAN Alliance.

Główną tezą niniejszej rozprawy jest twierdzenie, że mechanizmy rozwiązywania konfliktów mogą znacząco poprawić wydajność, stabilność i niezawodność sieci O-RAN działających z konfliktującymi aplikacjami. Badania opierają się na założeniu, że bez skutecznych środków wykrywania i rozwiązywania konfliktów korzyści płynące z otwartego ekosystemu nie mogą zostać w pełni wykorzystane. W związku z tym głównym celem badań jest zaproponowanie, wdrożenie i ocena kompleksowych ram (frameworku) wykrywania i rozwiązywania konfliktów, które byłyby uniwersalne, solidne i w pełni kompatybilne z architekturą O-RAN. Obejmuje to zaprojektowanie frameworku, który wpisuje się w fundamenty opisane w najnowszych standardach, a jednocześnie pozwala na rozwiązywanie wszystkich typów konfliktów. Badania mają na celu zdefiniowanie ram wykrywania i rozwiązywania konfliktów do wykorzystania zarówno w Near-RT RIC, jak i Non-RT RIC, zapewniając skuteczną neutralizację konfliktów w każdym możliwym scenariuszu sieciowym.

Metodologia przyjęta w niniejszej rozprawie opiera się na systematycznej strukturze, rozpoczynającej się od gruntownego przeglądu literatury, koncentrującego się na standardach O-RAN, raportach technicznych oraz pracach badawczych dotyczących koordynacji funkcji SON, w celu oceny stanu wiedzy. Na podstawie zidentyfikowanych luk badawczych, a w szczególności braku szczegółowej logiki rozwiązywania konfliktów, badania przeszły do etapu konceptualizacji i projektowania ramowego systemu rozwiązywania konfliktów nazwanego Conflict Mitigation Framework (CMF). CMF został zaprojektowany jako proceduralne i agnostyczne pod względem logiki rozwiązanie do wykrywania i rozwiązywania konfliktów w sieciach O-RAN. Aby ocenić korzyści płynące z tego rozwiązania, w ramach badań zaprojektowano i wdrożono autorskie środowisko symulacyjne sieci O-RAN. Oprogramowanie to modeluje środowisko sieci i pozwala na realizację scenariuszy konfliktowych w spójnych warunkach, umożliwiając rzetelny pomiar metryk wydajnościowych (ang. Key Performance Metrics, KPIs).

W ramach proponowanego rozwiązania zaimplementowano wiele algorytmów wykrywania i rozwiązywania konfliktów w celu oceny ich skuteczności. Początkowe etapy koncentrowały się na implementacji konwencjonalnych metod rozwiązywania konfliktów, które posłużyły jako punkty odniesienia (baseline). Następnie badania objęły projektowanie i rozwój rozwiązania opartego na Sztucznej Inteligencji (SI), wykorzystującego konkretnie sztuczne sieci neuronowe (ang. Artificial Neural Networks, ANN) typu aktor-krytyk (ang. actor-critic) trenowane za pomocą uczenia ze wzmocnieniem (ang. Reinforcement Learning, RL). Podejście to pozwala na adaptacyjne, oparte na danych zarządzanie wykrywanymi konfliktami, zdolne do rozwiązywania złożonych konfliktów sterowania między aplikacjami. Wydajność tych algorytmów została przeanalizowana przy użyciu nowatorskiej metryki ewaluacji, zaproponowanej specjalnie do porównywania rozwiązań dla rozwiązywania konfliktów.

Poza symulacją, rozprawa weryfikuje praktyczność proponowanego rozwiązania poprzez eksperyment przeprowadzony z wykorzystaniem rzeczywistej sieci O-RAN. CMF został zaimplementowany i przetestowany w jednym z laboratoriów certyfikowanych przez O-RAN Alliance dla testów sieci O-RAN, nazywanych O-RAN Open Testing and Integration Center (OTIC). W środowisku testowym wykorzystano rzeczywisty sprzęt i oprogramowanie zgodne ze standardem O-RAN, co zapewnia, że proponowane rozwiązania są skuteczne nie tylko w symulacjach teoretycznych, ale także w rzeczywistej infrastrukturze sieciowej. Ta metodologia testowania, obejmująca projekt teoretyczny, symulację i walidację sprzętową, potwierdza tezę, że solidne rozwiązywanie konfliktów jest kluczowa dla niezawodności sieci O-RAN.

Wyniki badań dostarczają szczegółowego opisu sposobów wykrywania i rozwiązywania konfliktów, wypełniając lukę w obecnych standardach O-RAN. Proponowane rozwiązania stanowią jedne z pierwszych opracowań w obszarze O-RAN poświęconych temu kluczowemu zagadnieniu. Dostarczając wzorca zapewnienia integralności sieci O-RAN w środowisku ze sprzętem i oprogramowaniem od wielu dostawców, praca ta wnosi istotny wkład w rozwój nowoczesnych sieci RAN. Przeprowadzone badania mogą pomóc w określeniu sposobu rozwiązywania problemów z konfliktami sterowania we wszystkich sieciach O-RAN, potencjalnie wpływając na sposób optymalizacji przyszłych sieci RAN i zapewniając, że elastyczność operacyjna O-RAN nie zostanie osiągnięta kosztem stabilności.

Glossary

3GPP Third Generation Partnership Project

5QI 5G QoS Identifier

A2C Advantage Actor-Critic

AC Admission Control

AI Artificial Intelligence

AI-RAN Artificial Intelligence RAN

AMF Access and Mobility Management Function

ANN Artificial Neural Network

API Application Programming Interface

ATE Average Treatment Effect

BLER Block Error Rate

BRA Backhaul Resource Allocation

BS base station

CAN Cognitive Autonomous Network

CATE Conditional ATE

CB Call Blockage

CBM Cognitive Bargaining Mechanism

CCC Cell Configuration and Control

CCG Concurrent Cooperative Games

CCI Commonwealth Cyber Initiative

CCKE Counterfactual Conformal KPI Estimation

CCO Coverage and Capacity Optimization

CD Conflict Detection

CF Cognitive Function

CIO Cell Individual Offset

CKE Conformal KPI Estimation

CLI Command Line Interface

ConMit Conflict Mitigation

CMF Conflict Mitigation Framework

COC Cell Outage Compensation

COMIX generalized Conflict Management scheme for Multi-Channel Power Control in O-RAN xApps

CQI Channel Quality Indicator

CR Conflict Resolution

C-RAN Centralized RAN

CSI Channel State Information

C-SON Centralized SON

CSS Concurrent learning with Spatial Separation

CSV Comma-Separated Values

CU Centralized Unit

DAG Directed Acyclic Graph

DCD Direct Conflict Detection

DDQN Double DQN

DL Downlink

DME Data Management and Exposure

DNN Deep Neural Network

DQN Deep Q-Learning Network

DRL Deep RL

DRM Data Rate Maximization

D-SON Distributed SON

DU Distributed Unit

E2AP E2 Application Protocol

E2SM E2 Service Model

E2SM-RC E2 Service Model-RAN Control

EDGE Enhanced Data rates for GSM Evolution

EE Energy Efficiency

EG Eisenberg-Gale

HSPA+ Enhanced HSPA

EI Enrichment Information

eMBB Enhanced Mobile Broadband

ES Energy Saving

FL Federated Learning

FML Federated Meta-Learning

GCN Graph Convolutional Network

GNN Graph Neural Network

GPRS General Packet Radio Service

GRACE Graph-based xApps Conflict and Root Cause Analysis Engine

GSMA GSM Association

HDBSCAN Hierarchical Density-Based Spatial Clustering of Applications with Noise

HOM handover margin

HOO Handover Optimization

HRL Hierarchical RL

HSCSD High Speed Circuit Switched Data

H-SON Hybrid SON

HSPA High Speed Packet Access

IBN Intent-Based Network

IBNM Intent-Based Network Management

ICD Indirect Conflict Detection

ICIC Inter-Cell Interference Coordination

ImCD Implicit Conflict Detection

IO Intelligent Orchestration

KL Kullback-Leibler

KPI Key Performance Indicator

KPM Key Performance Measurement

KSBS Kalai-Smorodinsky Bargaining Solution

LLC Lower Layers Control

LLM Large Language Model

LMO Local Multi-Objective

LTE Long Term Evolution

MAC Medium Access Control

MIMO Multiple Input Multiple Output

ML Machine Learning

MLB Mobility Load Balancing

MLP Multilayer Perceptron

mMIMO Massive MIMO

MNO Mobile Network Operator

MRO Mobility Robustness Optimization

nABS number of Almost Blank Subframes

NBS Nash Bargaining Solution

NCCKE Naïve CCKE

Near-RT Near-Real Time

NEM Network Empowerment Mechanism

NF Network Function

NFV Network Function Virtualization

NI Network Interface

Non-RT Non-Real Time

NR New Radio

NSGA-II Non-dominated Sorting Genetic Algorithm II

NSSI Network Slice Subnet Instance

NSWF Nash Social Welfare Function

O-RAN SC O-RAN Software Community

OAM Operations, Administration and Management

OCC Optimistic Concurrency Control

O-CU Open CU

O-DU Open DU

ONAP Open Network Automation Platform

O-RAN Open RAN

O-RU Open RU

OS Operating System

OSS Operations Support System

OTIC Open Test and Integration Center

PDCP Packet Data Convergence Protocol

PG Parameter Group

PHY Physical

PMon Performance Monitoring

PPO Proximal Policy Optimization

PRB Physical Resource Block

QACM QoS-Aware Conflict Mitigation

QoE Quality of Experience

QoS Quality of Service

RAN Radio Access Network

RAT Radio Access Technology

ReLU Rectified Linear Unit

RET Remote Electrical Tilt

RF Radio Frequency

RIC RAN Intelligent Controller

RL Reinforcement Learning

RLC Radio Link Control

RLF Radio Link Failure

RRC Radio Resource Control

RS RAT Selection

RSRP Reference Signal Received Power

RT Real Time

RU Radio Unit

SC Service Continuity

SCS Subcarrier Spacing

SD Standard Deviation

SDL Shared Data Layer

SDR Software-defined Radio

SEBS Shannon Entropy Bargaining Solution

SHAP Shapley Additive Explanations

SINR Signal to Inference plus Noise Ratio

SL Supervised Learning

SLA Service Level Agreement

SME Service Management and Exposure

SMO Service Management and Orchestration

SMS Short Message System

SOM Self-Organizing Maps

SON Self-Organizing Network

srsRAN Software Radio Systems RAN

SVR Support Vector Regression

TE&IV Topology Exposure and Inventory Management

TL Team Learning

TM Topology Management

TS Traffic Steering

TSL Temporal Separation during Learning

TTT time-to-trigger

UE User Equipment

UM User Management

UMAP Uniform Manifold Approximation and Projection

UMF Unified Management Framework

UMTS Universal Mobile Telecommunications System

UPF User Plane Function

URLLC Ultra-Reliable Low-Latency Communication

USL Unsupervised Learning

USRP Universal Software Radio Peripheral

V2X Vehicle-to-Everything

VM Virtual Machine

vRAN Virtualized RAN

WCP Weighted Conformal Prediction

WG Working Group

WLAN Wireless Local Access Network

WNBS Weighted NBS

Chapter 1

Introduction

Instant wireless access to data has become a taken-for-granted privilege for people in digitally advanced countries. It provides the means to consume and share media, gather information, access services, work, and socialize, in virtually any place. This applies to the vast majority of adults in Europe and United States of America. Over 86% inhabitants of countries comprising the European Union aged 16-74 claim to access Internet via mobile phone [8], while over 40% of adult Americans claim to be constantly on-line and 90% claim to access Internet at least once a day [9].

Such impactful and dependable connectivity service would not be possible without the decades of advancements in key research fields—ranging from electronics and antenna design to signal processing and computing. These developments have lead to establishment of robust connectivity solutions, which enable common mobile access to Internet.

Radio Access Network (RAN) is one of the key enablers of ubiquitous connectivity that the modern societies have learned to depend on. It started as an analog network based on large cells and allowed its users to make calls without wired connection to the phone. Adoption of this initial generation of RAN was the first major step towards establishing pervasive connectivity. Over the years, architecture and technology behind RAN have gone through multiple transformations, including switch to being digital and providing new value-added services to its users. These new services include these requiring mobile access to Internet, such as video streaming, cloud storage, social media, and so on.

As the RAN services based on Internet access become more common and, simultaneously, fidelity and volume of media transferred while providing these services increase, the available throughput of data provided by RAN needs to increase accordingly. A report by GSM Association (GSMA) forecasts a 300% increase in global demand for mobile data transfer between 2024 and 2030 [10]. The observed trend is that this exponential growth of mobile data demand has been stable in the recent years [10–12]. These market conditions are driving the Mobile Network Operators (MNOs) to expand their network infrastructure to accommodate the growing traffic by increasing the total capacity of the RAN. Simultaneously, this demand is also addressed by RAN vendors and researchers in terms of the required technological advancements to increase efficiency of radio transmission and management of limited radio resources.

Evolution of RAN, driven by the increasing demand on mobile data traffic, could not be possible without improvement of its computing capabilities and deployment flexibility. Recent trends include introduction of Artificial Intelligence (AI) and Machine Learning (ML) support into RAN architecture to enable complex computing of large volumes of data within restricted time frames [13]. Another trend is cloudification, i.e., implementing RAN software as microservices and

decoupling it from proprietary hardware, enabling deployment on general-purpose infrastructure. This approach allows for increased programmability and scalable resource management [14].

One of the challenges that MNOs face when expanding their RAN is the closed design of the conventional RAN equipment [15]. In these deployments, each component in the chain is a black box (i.e., hardware and software with opaque internals) produced by one of the few vendors present in the market. Most often, such component can be integrated only with components provided by the same vendor. Programmability of these components is up to the vendor and can be heavily restricted. The closed design of conventional RAN equipment also naturally leads to vendor lock-in in the MNO's network: as some components can be integrated only with others produced by the same vendor, MNOs are limited in terms of choice in equipment for network expansion. Vendors, aware of these limitations, are in a beneficial position over their customers when negotiating deals with MNOs.

O-RAN is a concept of bringing openness, intelligence, and virtualization into otherwise closed RAN. The O-RAN Alliance [16] is the main standardization body for this cause. It was formed in 2018 by a consortium of MNOs, research institutions, and RAN vendors, expanding over the years to reach over 260 member companies in 2025.

O-RAN is based on disaggregated virtualized network functions with open interfaces in between. Hence, in contrast to the conventional RAN, O-RAN allows the MNOs to deploy base stations with combinations of hardware and software components provided by various vendors. This difference is depicted in Figure 1.1.

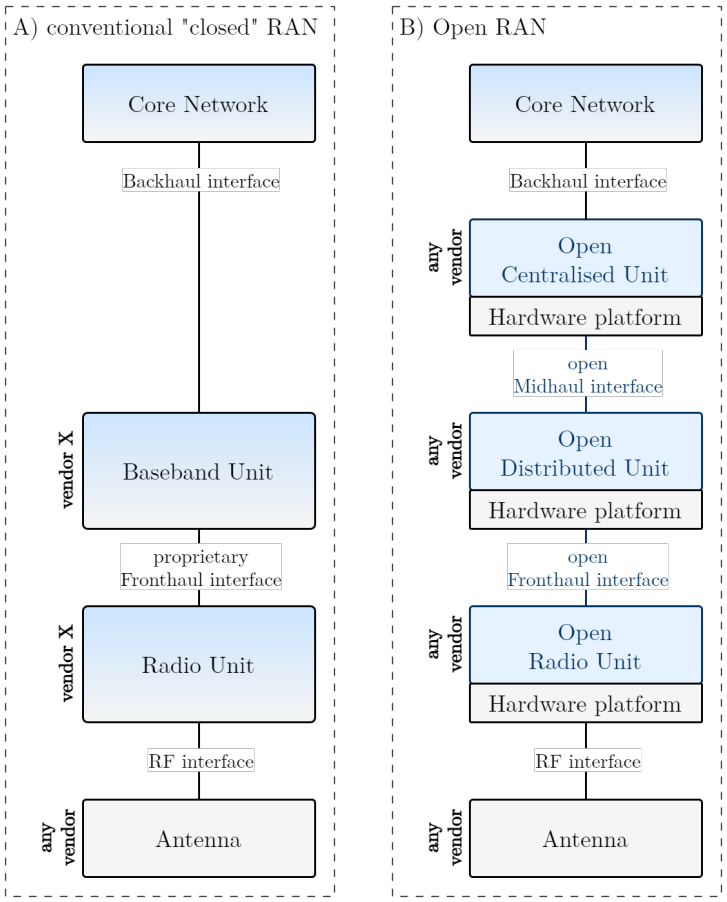

FIGURE 1.1: Comparison of conventional closed RAN (A) and Open RAN (B)

Source: own elaboration

Enabling flexibility in choice of RAN hardware and software components is a crucial benefit for MNOs. Furthermore, O-RAN provides uniform programmability of the components, which in case of conventional RAN could be restricted by the vendor. The split of disaggregated network functions in O-RAN depicted in Figure 1.1 allows for distribution of specific network functions in various distances from the antenna: Open RU (O-RU) close to the mounting point of the antenna (or integrated within the antenna), Open DU (O-DU) on the same site nearby the antenna, and Open CU (O-CU) in the cloud edge. Depending on the deployment scenario, many network functions can be hosted on the same hardware platform. For example, O-DU and O-CU can be hosted together in a scenario similar to conventional RAN's Baseband Unit.

Other than openness and virtualization, O-RAN aims to introduce robust intelligence for optimization of RAN operation. This is implemented by introduction of RICs—dedicated logical components for processing information about network conditions and computing control decisions for optimization of the network. The RICs are used to host various applications, which provide value-added services. This gives the MNO the possibility to customize the feature-set of the RAN intelligence deployed in O-RAN.

The unprecedented programmability of RAN enabled by O-RAN introduces multiple management challenges that MNOs need to deal with. The multi-vendor mix of hardware and software components requires complex orchestration not only to prevent interoperability issues where possible, but also to detect and mitigate them efficiently if they happen in live network. This challenge is even more difficult as the cloud-native O-RAN allows to deploy the network functions across multiple distributed locations, making robust network infrastructure monitoring measures even more crucial. Furthermore, integrating components from multiple vendors raises additional security concerns and MNOs need to ensure the integrity and trustworthiness of each deployed component. Another challenge is related to ensuring that the deployed RAN intelligence measures are fed the required data in a reliable and timely manner, as any disruptions in the data processing pipelines may cause downtime of the deployed applications or lead to unexpected behavior.

There are multiple actors within the O-RAN architecture which are capable of executing control decisions. Namely, the RICs along with deployed applications. As these entities operate simultaneously and may target various optimization goals, there is a high probability that control conflicts will happen during operation. Mitigating these conflicts is crucial to ensure reliable and efficient operation of the system. Similar challenges are present within other complex computer architectures. Some examples include a PC executing concurrent tasks competing for CPU compute time, or multiple applications in an Android smartphone requesting to handle an event raised in the Operating System (OS). In context of RAN, similar considerations regarding control conflicts were a part of developing Self-Organizing Networks (SONs), where diverse radio optimization algorithms can be concurrently executed in the RAN. The RAN optimization logic in SON can be distributed across the base stations or centralized in the Operations, Administration and Management (OAM) component of the network. This is in contrast to O-RAN, where the RAN intelligence logic is delegated to RICs. Hence, the considerations from SON cannot always be applied directly to O-RAN.

The area of ConMit for O-RAN RICs has been envisioned within the architecture of the system as dedicated ConMit functionality blocks within the RICs. Other than outlining the basic goal of the ConMit components, the O-RAN specifications do not specify any detailed logic on how to mitigate the conflicts. In October 2024, O-RAN Alliance published a dedicated technical report investigating the issue of ConMit in O-RAN [17]. The report describes key issues for detection, resolution, and avoidance of conflicts, and proposes procedural solutions for these activities. However, this report does not provide guidance on detailed logic on how to compute

an optimal conflict resolution. Separate from the ConMit report published by O-RAN Alliance, there are many research publications regarding this topic. As an example, paper published in 2023 by the author described challenges and potential solutions for ConMit in O-RAN along with categorization of ConMit activities [1] before publication of the O-RAN report. There are also multiple papers describing various examples of ConMit tactics [2][18–24].

1.1 Thesis, motivation, and goal of the dissertation

The central thesis of this dissertation is as follows:

> **Conflict mitigation mechanisms can significantly improve the performance, stability, and reliability of O-RAN networks operating with conflicting applications.**

This thesis originates from the observation that the open and programmable nature of O-RAN, while enabling flexible and intelligent RAN intelligence, inherently increases the likelihood of control conflicts between independently developed applications running in parallel. As such, effective detection and resolution of such conflicts becomes essential to ensure consistent network operation and allow MNOs to fully exploit the benefits of O-RAN.

The primary goal of this dissertation is to design, implement, and evaluate a conflict mitigation framework that is fully compatible with the O-RAN standard, is adaptable to diverse application configurations, and substantially improves the network operation through intelligent control resolution mechanisms.

1.2 Research methodology

Methodology of the research in this dissertation was comprised of several subsequent phases to systematically address the problem of conflict mitigation in O-RAN.

The research process is started with a thorough literature review to assess the state of the art in conflict mitigation in O-RAN, including legacy solutions developed for SON. The review included a wide range of publications, varying from specifications issued by O-RAN Alliance and Third Generation Partnership Project (3GPP), through technical reports and white papers, to academic publications. This phase identified key research gaps, particularly the lack of robust conflict resolution methods fully compatible with O-RAN standards.

Building on the lessons learned from the review, the subsequent phases focused on identifying and analyzing the key challenges associated with effective ConMit in O-RAN, followed by designing a framework for ConMit fitted into to the existing O-RAN architecture. This framework takes into consideration specific use cases and limitations resulting from the O-RAN standards and allows to host any conflict detection and resolution logic to mitigate any conflicts between xApps in Near-RT RIC.

Two basic conflict resolution methods were first implemented to assess the benefits of implementing ConMit according to the proposed framework. The later stages of the study focused on the design of a conflict resolution algorithm employing Reinforcement Learning (RL) to effectively resolve a control conflict with any conflicting applications.

Each proposed conflict resolution algorithm was evaluated using a simulation-based experimental setup emulating a representative O-RAN environment. The simulation runs enable measurement of Key Performance Indicators (KPIs) in consistent conditions, allowing for a fair comparison of ConMit methods. Analysis of the results was performed using a novel metric for comparison

of conflict resolution solutions proposed by the author. The proposed RL-based approach was evaluated with the two basic solutions as baselines.

The described methodology ensured that the proposed ConMit solution was developed based on well-established foundations and, in result, allowed for rigorous testing of the research thesis.

1.3 Structure of the dissertation and main contributions

Chapter 1 is the introduction to the dissertation. It outlines the context relevant to the topic of ConMit in O-RAN and argues for the relevancy of the considered research topic. Furthermore, it describes the research thesis, methodology, and structure of the dissertation.

Chapter 2 considers the topic of intelligence in RAN. It starts with a brief summary of the evolution history of RAN leading to the relevant concept of O-RAN, pinpointing the key improvements in each generation of RAN and identifying how robust intelligence measures enable improvement of RAN performance. Further sections of this chapter describe the framework for RAN intelligence enabled by RICs in O-RAN, identifies various types of applications in O-RAN, and outline how AI/ML workflows are envisioned in O-RAN RICs. The last section of this chapter provides a look into the future of RAN intelligence with the conception of the AI-RAN Alliance and summarizes its main objectives.

Chapter 3 describes the field of conflict management in RAN. First, a general view of conflicts in O-RAN is provided to establish a baseline understanding of what control conflicts need to be considered by ConMit activities. Next, challenges for ConMit in O-RAN are identified and relevant guidelines for ConMit solutions are provided. This chapter concludes with a summary of the existing ConMit solutions described in the available research papers and articles.

Chapter 4 showcases the proprietary simulation environment implemented by the author for the purpose of evaluating proposed ConMit measures. First, the graphical user interface of the simulation software is presented with screenshots and descriptions of each interface element. Then, the implemented model of an O-RAN environment is detailed, along with the flow of the simulation process. Finally, the information on modes of operation supported by the simulation software is provided.

Chapter 5 contains the descriptions of the O-RAN ConMit approach proposed by the author. Categorization of ConMit activities is provided to identify the possible approaches for ConMit. Then, CMF is presented as a procedural, logic-agnostic solution to detection and resolution of all types of control conflicts in O-RAN. Furthermore, the proposed conventional conflict detection and mitigation algorithms are presented, along with evaluation results. Evaluation of Conflict Resolution (CR) methods is performed using a novel evaluation approach proposed by the author.

Chapter 6 addresses the AI/ML solutions for ConMit proposed by the author. The general utility of AI/ML in O-RAN ConMit is outlined. Then, a solution for conflict resolution using Artificial Neural Network (ANN) trained using RL is described, along with evaluation results.

Chapter 7 presents the results of an evaluation of CMF in an O-RAN Open Test and Integration Center (OTIC), a laboratory certified by the O-RAN Alliance. The testbed runs real O-RAN-compliant hardware and software.

Chapter 8 concludes the dissertation by summarizing the described work and listing the key takeaways. The possible research directions for the future work are also proposed.

The main contributions by the author are listed in Table 1.1.

Table 1.1: Main contributions and their corresponding chapters

No.	Main contribution	Chapter reference(s)
1	Analysis of the evolution of RAN towards O-RAN, identifying key architectural shifts and describing the specific features of RICs that enable robust, intelligent control of the network, setting the context for algorithmic optimization.	2
2	Identification and analysis of key challenges for ConMit in O-RAN, including reliability of detection, maintenance in evolving networks, and observability limits. This analysis establishes the motivation and requirements for the proposed solutions.	3
3	Comprehensive survey of the state of the art in ConMit for O-RAN, distinguishing it from legacy SON approaches. The survey identifies specific research gaps, particularly the lack of logic-agnostic frameworks compatible with standard O-RAN interfaces.	3
4	Novel categorization of ConMit activities in O-RAN into Preventive ConMit, Conflict Detection and Resolution (CD&R), and Supervision and Adaptation (S&A).	3
5	Design and implementation of a proprietary, discrete-event O-RAN network simulation environment. The tool is specifically tailored for the evaluation of ConMit algorithms, featuring custom modules for xApp logic, RIC agents, and flexible logging of conflict events.	4
6	Conceptualization and design of the Conflict Mitigation Framework (CMF) as a modular extension to the Near-RT RIC. The framework introduces dedicated ConMit components and defines message flows for mitigating conflicts without requiring modifications to third-party xApps.	5
7	Definition of a comprehensive methodology for evaluating CR methods, introducing a novel weighted penalty metric that aggregates negative network events to quantify the effectiveness of CR methods.	5
8	Implementation and evaluation of rule-based algorithms (prioritization and prioritization with cooldown) within the CMF. These serve as baselines for evaluating CR methods.	5
9	Development of ACCoRD, an ANN-based CR method utilizing RL via PPO-Clip algorithm. This contribution demonstrates how AI-driven CR methods can dynamically adapt resolution policies to minimize penalties in complex O-RAN environments.	6
10	Comparative evaluation of implemented ConMit solutions, proving the positive impact of ConMit methods and providing empirical evidence that, while rule-based methods suffice for simple conflicts, AI-driven approaches significantly outperform baselines in dense deployment scenarios.	5, 6
11	Experimental validation of the proposed CMF in a real-world O-RAN OTIC. This contribution provides the first known hardware-based evaluation of ConMit using OTA transmission, proving the framework's capability to stabilize network throughput in live operations.	7

1.4 Declaration on using generative AI

During the preparation of this work, the author used ChatGPT and Google Gemini for proofreading and Writefull for editing and grammar enhancements. This work (text, contents, figures and results) are the result of the author's work and were not created using generative AI.

Chapter 2

Intelligence in RAN

2.1 Evolution of RAN towards O-RAN

2.1.1 RAN generations

The first generation of analog RAN launched in 1979 and enabled wide-area mobile voice with low-cost devices [13]. Progression of RAN lead to its second iteration called 2G, marking a switch to digital radio with circuit-switched voice, Short Message System (SMS), and early data via High Speed Circuit Switched Data (HSCSD), later upgraded to packet-switched data via General Packet Radio Service (GPRS)/Enhanced Data rates for GSM Evolution (EDGE) [25, 26]. The 3GPP consortium was created in early 2000s to steer specification of 3G RAN called Universal Mobile Telecommunications System (UMTS); it increased data transfer rates to enable multimedia services, with further improvements included in High Speed Packet Access (HSPA)/Enhanced HSPA (HSPA+) [27–29]. Rising performance targets caused the need to develop 4G Long Term Evolution (LTE) RAN, including all-IP core and voice, with higher spectral efficiency via Multiple Input Multiple Output (MIMO) and advanced modulation schemes [26].

As of 2025, the latest deployed generation of RAN is 5G, which is based on the New Radio (NR) air interface. In comparison to its predecessors, 5G supports more flexible radio channel configurations, higher data rates, and lower latencies [30]. One of the means to meet these targets is Massive MIMO (mMIMO) [30, 31], improving on the antenna technology initially utilized in LTE. Furthermore, 5G improves on the RAN programmability by introducing network slicing–splitting the network resources into separate end-to-end logical networks [32].

As part of work on the fifth generation of RAN, 3GPP studied the possible functional split options in logical components comprising a gNodeB (a 5G base station (BS)) [33]. The functional splits describe where the boundary in protocols handled by separate logical components comprising a gNodeB is. Out of eight possible options, 3GPP standardized option 2 for split between the Centralized Unit (CU) and the Distributed Unit (DU) [34, 35]. In this split, CU hosts L3 (i.e., Radio Resource Control (RRC)) and L2 (Packet Data Convergence Protocol (PDCP)) protocols. The CU is also decomposed into two logical components handling control and user planes, named CU-CP and CU-UP respectively. The DU connects to the CU via the F1 interface and, in the 3GPP 5G specification, is responsible for handling the remaining L2 and L1 protocols. In contrast, O-RAN standard introduces further detailed split but combining the 5G option 2 CU-DU split with an additional split according to option 7 to extract low part of Physical (PHY) and Radio Frequency (RF) L1 protocols into Radio Unit (RU). This leaves Radio Link Control (RLC) (L2), Medium Access Control (MAC) (L2) and high part of PHY (L1) in the DU, creating the 7-2x functional split. The functional 7-2x split is shown in Figure 2.1.

Work on the next generations of RAN (6G and beyond) is currently in progress. The sixth generation is still being defined, but the main directions include addressing the long-term challenges which are still unaddressed by the previous generations, utilizing broader bandwidths for further increase in throughput (including terahertz bands), reduce capital and operating expenditures related to managing the network, and becoming even more user-centric [13, 36].

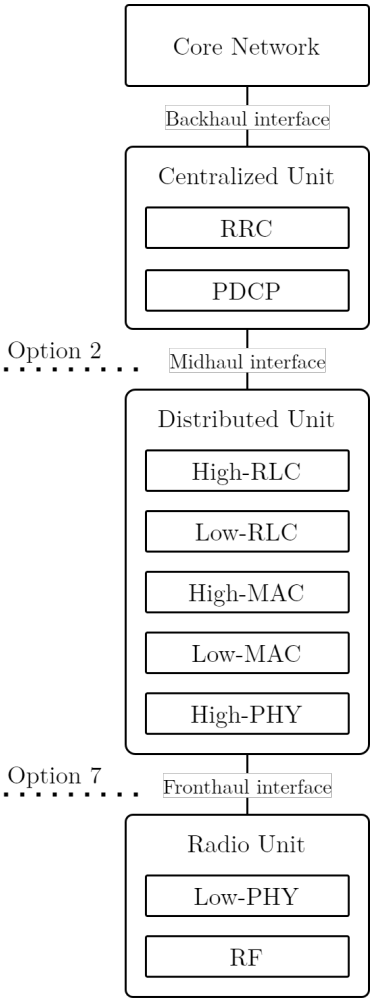

FIGURE 2.1: 7-2x functional split

2.1.2 Self-Organizing Networks

SON was introduced as part of LTE specifications in 2009 with the goal to improve operability in multi-vendor RAN deployments via interoperable RAN intelligence measures [37]. The concept of SON also applies to 5G [38]. To ensure interoperability in SON, the scope and format of information exchanged between RAN nodes in SON is standardized. The following functionalities are envisioned for SON [37]:

- self-healing - automated recovery from network outages based on fault information,
- self-optimization - automated optimization of RAN setup, e.g., with regards to coverage, energy usage, inter-cell interference,
- self-configuration - automated configuration of RAN infrastructure and relevant connectivity.

Other than these basic use cases, SON envisions supporting functions, including minimization of drive tests and coordination function for mitigation of conflicts [37] (often called self-coordination in literature).

Network control logic in SON can be spread across network elements. Specifically, the following SON architecture approaches can apply [38]:

- Centralized SON (C-SON) - SON control logic is managed and executed from within the OAM component of the network. This approach requires additional interface to RAN nodes to communicate network control messages to specific base stations, but guarantees configuration simplicity and consistent control across all RAN nodes.
- Distributed SON (D-SON) - SON control logic is locally deployed in RAN nodes. Coordination of reconfiguration activities requires information exchange through additional interface between the RAN nodes. This method is prone to conflicts, as the control decisions are not centrally managed. Methods of control can differ between RAN nodes, leading to inconsistency in their operation. Management role of OAM system is limited to enabling/disabling SON functions and providing supplementary information.
- Hybrid SON (H-SON) - combination of centralized and distributed SON approaches, where both are applied and various use cases are realized centrally in OAM and in specific RAN

nodes. This approach combines advantages of the two approaches, but requires more effort to reliable deploy and coordinate.

These three SON architecture approaches are shown in Figure 2.2.

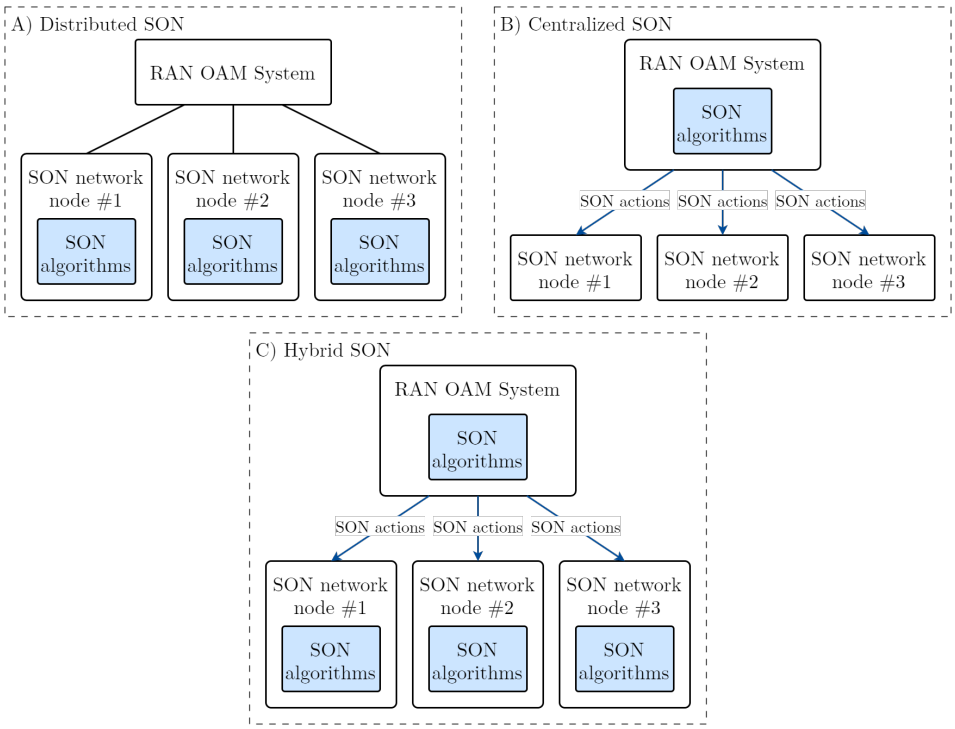

FIGURE 2.2: SON architecture approaches

Source: own elaboration based on [38]

Several key use cases for SON in 5G are identified in technical specifications defined by 3GPP [38]:

- Random Access Optimization - in D-SON, automatic configuration of parameters of random access channel to reduce network access time and minimize access failures,
- Mobility Robustness Optimization (MRO) - in D-SON, automatic configuration of handover parameters to improve handover performance,
- Load Balancing Optimization - in D-SON and C-SON, automatic redistribution of traffic among nearby cells to ensure efficient usage of radio resources while retaining network performance and user experience,
- Capacity and Coverage Optimization - in C-SON, automatic configuration of cell parameters (such as transmission power, antenna tilt) to optimize radio resource usage and user experience.

2.1.3 Flexible RAN deployment architectures: C-RAN, vRAN, Cloud RAN, O-RAN

The fourth generation of RAN introduced increased flexibility in deployment of base stations in form of Centralized RAN (C-RAN) architecture [39]. This means that the baseband processing function of the RAN starting from 4G can be centralized for multiple base stations. For example, an MNO could handle traffic from multiple cell sites with baseband processing units located in a single data center location connected to the cell sites via fiber connectivity. This approach allows

for significant cost reduction by significantly decreasing the amount of hardware deployed on the cell site.

Another deployment architecture type growing in popularity is Virtualized RAN (vRAN) [39, 40]. It works by decoupling RAN software from hardware through the means of virtualization. In networking context, this approach is called Network Function Virtualization (NFV). From an MNO's operational perspective, it means that any server hardware can run compatible RAN software in a containerized manner. This furthers the flexibility of modern RAN and allows for robust automation of deployment activities.

The combination of C-RAN and vRAN is called Cloud RAN [39, 41]. It centralizes the baseband processing resources in a single physical location, while simultaneously applying NFV to deploy logical RAN components as microservices. This approach enables automated elastic network reconfiguration, such as healing or shifting resources between cells to balance load.

O-RAN, as specified by the O-RAN Alliance, builds directly on the virtualization baseline of Cloud RAN [39, 42]. In comparison to virtualized but potentially proprietary Cloud RAN, O-RAN enforces a fully disaggregated architecture by standardizing open interfaces between the virtualized Network Functions (NFs). This enables cross-vendor interoperability for truly open and agile RAN [43]. Architecture of O-RAN is envisioned to support future RAN generations. Furthermore, O-RAN defines additional logical components tailored to manage and execute RAN intelligence services, with native support of AI/ML workflows [44, 45]. This programmable intelligence represents a significant evolution of the automated RAN control functions previously specified in SON.

The high level logical architecture of an O-RAN system with both open interfaces defined by O-RAN Alliance (drawn in light blue) and 3GPP interfaces (drawn in black) is shown in Figure 2.3. The O-RAN architecture includes Open Fronthaul interface between the O-DU and O-RU as a supplement to 3GPP's interfaces between RAN components–namely, the F1-c and F1-u interfaces between the O-CU and O-DU, the D2 interface between peer O-DUs, the Xn and X2-c interfaces between peer O-CUs, and the NG-c and NG-u interfaces between the O-CU and the core network. Other than these intra-RAN interfaces, O-RAN introduces dedicated components enabling RAN intelligence along with the necessary interfaces: A1, O1, O2, Y1, and E2. These components and interfaces are described in more detail in Section 2.2.

By combining the principles of Cloud RAN and envisioning dedicated logical components for AI/ML-enabled RAN intelligence, O-RAN is considered one of enablers of innovations in 6G (and beyond) networks [14]. The standards defined by O-RAN Alliance serve as a cloudified deployment framework acting as a foundation to the baseline RAN functionalities defined by standardization groups such as 3GPP.

Architectures of conventional RAN, C-RAN, vRAN, and O-RAN deployments are compared in Figure 2.4 and Table 2.1 shows the typical features of each architecture type.

2.2 RAN Intelligent Controllers in O-RAN

One of the main promises of O-RAN is enabling robust RAN intelligence to effectively optimize complex networks. O-RAN architecture envisions dedicated NFs called RAN Intelligent Controllers (or RICs, in short) capable of hosting various applications providing added-value services in the network.

Intelligent RAN control in O-RAN is organized into three main control loops. Each loop differ with the applicable time scale. Namely, the following control loops are considered in O-RAN [43]:

- Non-Real Time (Non-RT) - over 1 second,

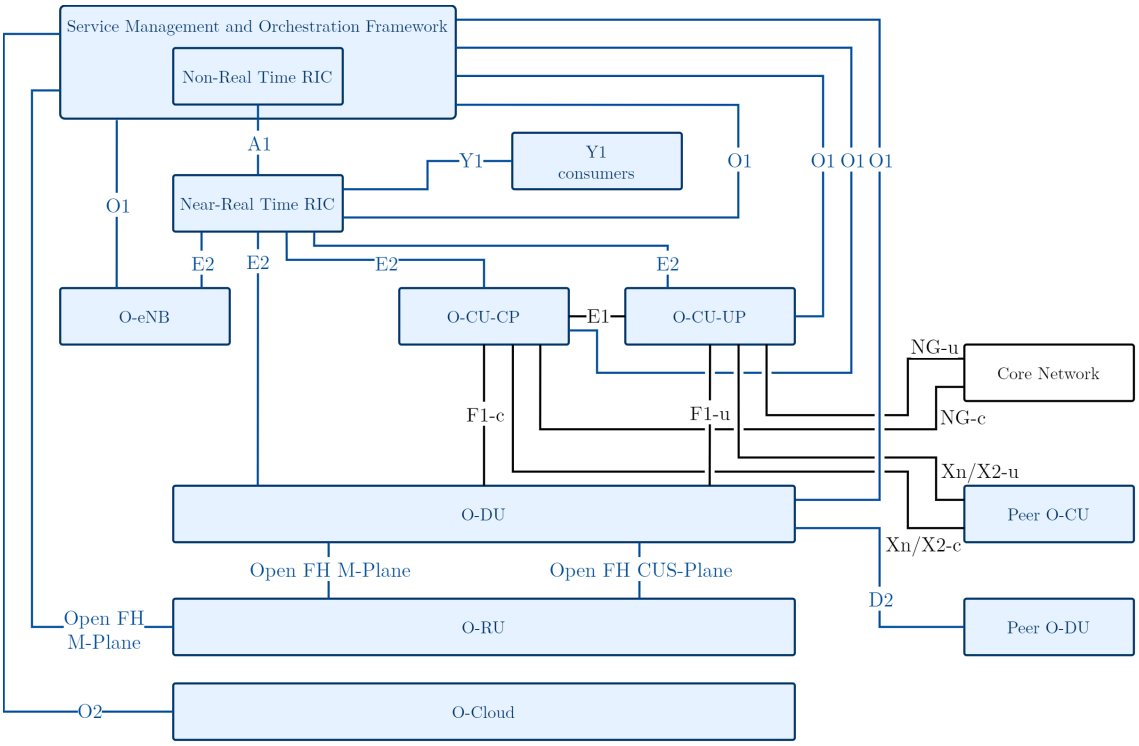

FIGURE 2.3: Logical O-RAN architecture with interfaces

Source: own elaboration based on [43]

Architecture	Summary	Hardware	Interfaces	Virtualization	Deployment type
conventional RAN	Each cell site integrates baseband and radio from a single vendor	proprietary	proprietary	not supported	distributed
C-RAN	Centralized pool of baseband units serves multiple cell sites	proprietary	proprietary	not supported	centralized
vRAN	Virtualized software running on (and decoupled from) proprietary hardware	COTS servers	proprietary	supported	centralized or distributed
Cloud RAN	Combination of C-RAN and vRAN - virtualization with cloud-native microservices and centralized pooled baseband processing	COTS servers, cloud platforms	proprietary	supported	centralized
O-RAN	Disaggregated and cloud-native multi-vendor RAN with open interfaces	any compatible	open	supported	centralized or distributed

TABLE 2.1: Comparison of typical features of RAN deployment architectures

Source: own elaboration based on [14, 39–42]

- Near-RT - between 10 milliseconds and 1 second,
- Real Time (RT) - under 10 milliseconds.

These control loops are operating simultaneously on various levels of the O-RAN architecture. In some cases, the control loops may interact with each other [43].

Control in the Non-RT control loop is strictly related to the Non-RT RIC, which is a part of the Service Management and Orchestration (SMO) Framework envisioned as part of the O-RAN architecture [44]. One of its primary purposes is to serve guidance and policy updates to the shorter control loops [46]. The Non-RT RIC is responsible for providing enrichment information and managing ML models for the Near-RT control loop. The Non-RT RIC communicates with the Near-RT RIC via the A1 interface. In addition to the A1 interface, SMO, which acts as the host of Non-RT RIC, communicates with O-RAN NFs via the O1 interface and with O-Cloud via

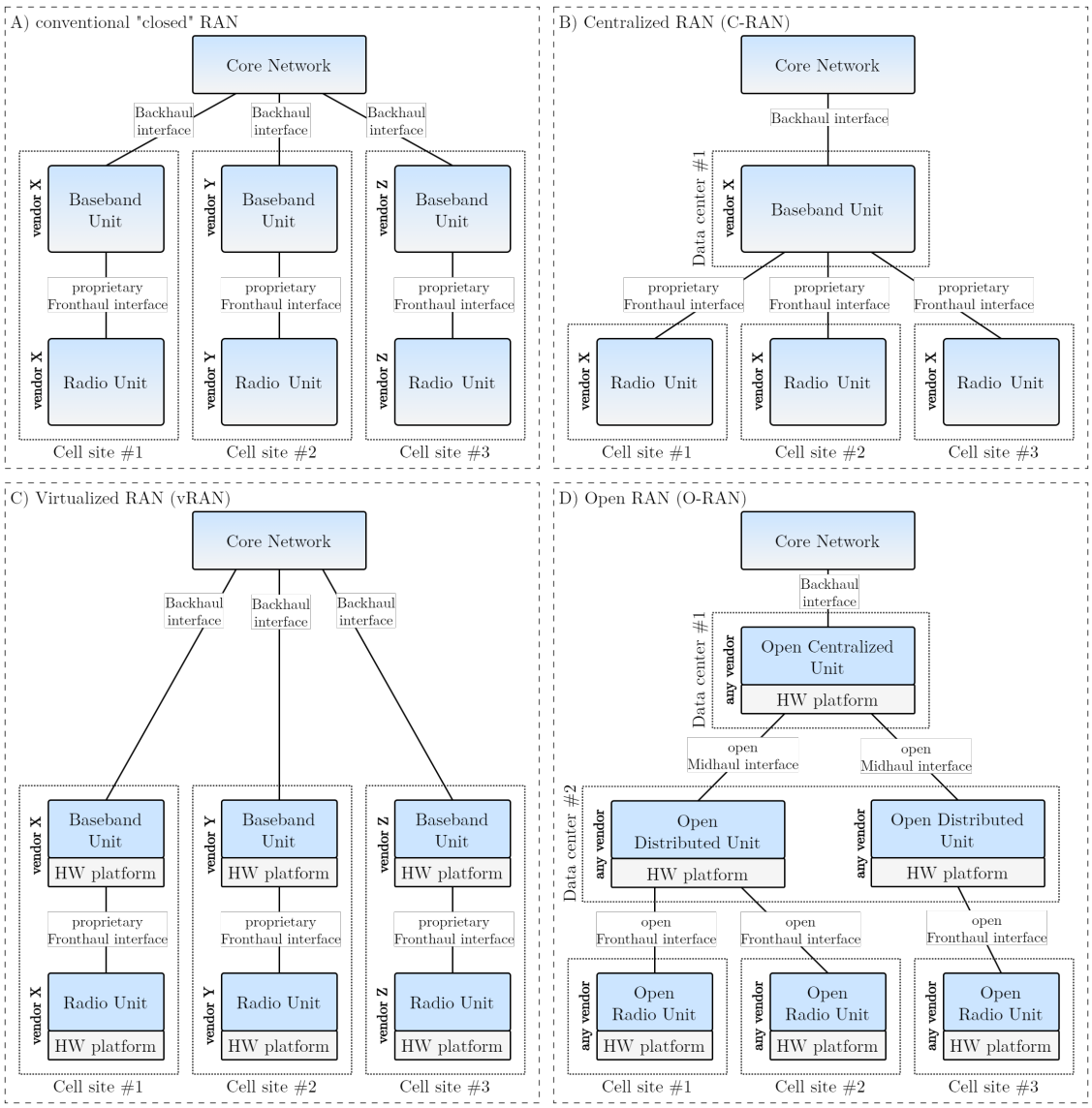

FIGURE 2.4: RAN deployment architectures

Source: own elaboration

the O2 interface. These interfaces are used in the Non-RT control loop to dynamically operate and maintain the network by reconfiguring NFs and performing software update and deployment activities [47, 48]. Management activities for the O2 interface include deployment and infrastructure management of the cloud resources [43].

As for the Near-RT control loop, Near-RT RIC is the component for hosting its RAN steering logic [45]. Near-RT RIC uses the E2 interface to RAN nodes and E2 Service Model (E2SM) to gather data about performance of RAN nodes, perform radio resource management, modify cell and lower layers configuration [49–51]. Network control actions performed by Near-RT RIC can be guided with policies and enrichment information provided by Non-RT RIC through the A1 interface. Furthermore, Near-RT RIC can use the available data to generate RAN statistics and may provide it to analytics consumers via the Y1 interface.

According to the O-RAN specifications, the RT control loop is executed directly in the 5G RAN nodes within the O-RAN architecture—in O-DU and O-RU. RT control actions include radio resource scheduling, modification of physical layer parameters and beam management [15]. O-RAN

specifications do not currently specify the RT control loops, but consider them for future study. However, existing research propose extending the RT control loop with additional logic in the form of dApps deployed in O-CU and O-DU [52].

O-RAN control loops and RICs are shown in Figure 2.5.

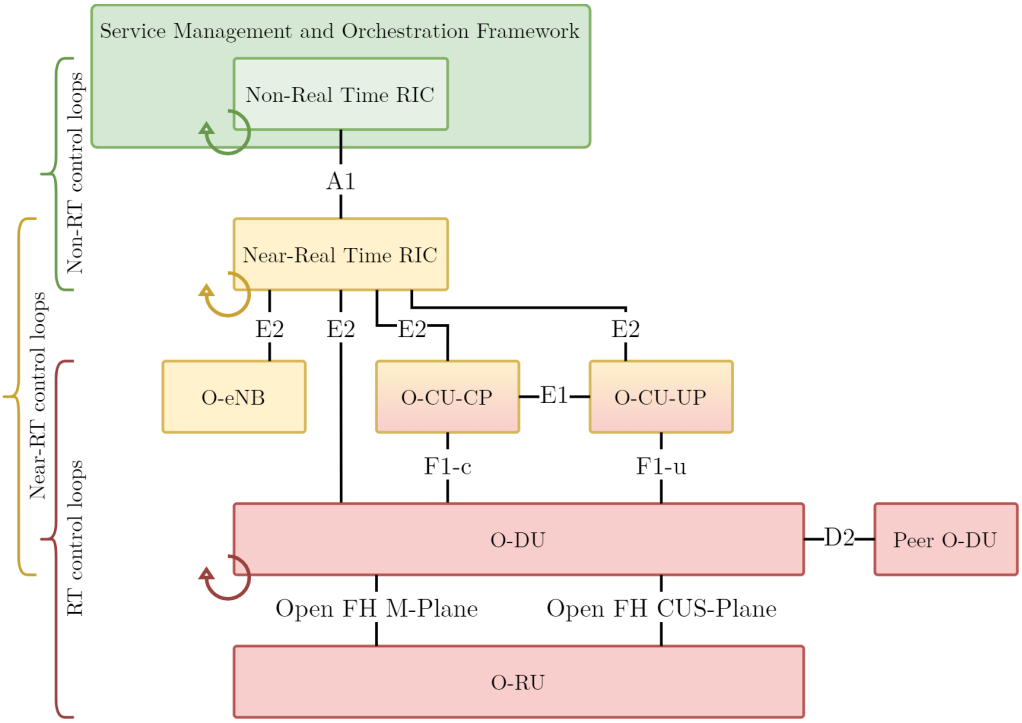

FIGURE 2.5: O-RAN control loops

Source: own elaboration based on [15, 43, 52]

2.3 rApps, xApps, dApps

Robust programmability and intelligence in O-RAN is achieved using applications, or apps in short. These apps are either hosted in the relevant RIC or operate alongside RAN nodes. The main goal of each app is to delivery specific functionality. O-RAN specifications describe rApps running in the Non-RT RIC [44] and xApps running in the Near-RT RIC [45]. An further extension to the O-RAN application landscape was proposed in [52, 53] with dApps that can run in the RT control loop in O-CUs and O-DUs.

2.3.1 rApps in Non-RT RIC

An O-RAN deployment architecture generally includes a single Non-RT RIC within the SMO Framework [43] and may run multiple rApps. The Non-RT RIC employs the R1 interface to rApps. This interface provides rApps with the following R1 services [44, 54]:

- rApp-related - allows rApps to provide manage and performance information,
- Service Management and Exposure (SME) - allows rApps to register, discover, authorize and authenticate services,
- Data Management and Exposure (DME) - allows rApps to collect and provide data,
- RAN-OAM-related - allows rApps to manage RAN nodes and query information about performance and faults of RAN nodes,

- A1-related - allows rApps to manage A1 policies,
- AI/ML-workflow-related - allows rApps to manage, train and deploy AI/ML models,
- Topology Exposure and Inventory Management (TE&IV) - allows rApps to access information about network topology and inventory.

Any rApp and the SMO framework may produce and consume zero or more R1 services [44].

The A1 interface can be used by rApps (which communicate with the Non-RT RIC via the R1 interface) to communicate A1 policies to Near-RT RICs as guidance for RAN optimization activities. Then, Near-RT RICs along with their applications act as policy enforcers. Furthermore, the A1 interface is used by the Non-RT RIC to communicate enrichment information provided by rApps towards Near-RT RICs. There is a number of A1 services provided by the SMO framework and the Non-RT RIC to rApps. These services include the following [44]:

- A1 services related to A1 policy management:
  - querying supported A1 policy types - includes discovery of supported A1 policy types and subscription to changes in SMO's list of supported A1 policy types,
  - managing A1 policies - includes all discovery and management activities related to A1 policies created by an rApp, i.e., querying, creating, updating, and removing; an rApp may also query A1 policies created by other rApps, if permitted by SMO,
  - querying enforcement status of A1 policies - includes discovery of enforcement status of A1 policies created by an rApp; an rApp may also query A1 policies created by other rApps, if permitted by SMO,
  - subscribing to event notifications related to A1 policies - includes management of subscriptions to notifications about events related to A1 policies (i.e., creation, update, removal) and enforcement status of A1 policies; subscription requests regarding A1 policies created by other xApps are accepted only if SMO permits it,
- A1 services related to A1 enrichment information:
  - managing A1 enrichment information types - includes registration and deregistration of enrichment information types which an rApp can produce; enrichment information type cannot be updated, so an rApp wanting to modify an existing registered A1 enrichment information type must deregister the existing type and register a new one.

2.3.2 xApps in Near-RT RIC

Concurrent operation of multiple Near-RT RICs is envisioned as part of O-RAN. Similarly to rApps in Non-RT RIC, each Near-RT RIC may host multiple various xApps. The Near-RT RIC hosts functionalities acting as the baseline for xApp operation [45]:

- Database and Shared Data Layer (SDL) - allow RIC to maintain and expose information about RAN nodes and User Equipments (UEs),
- xApp Subscription Management - allows RIC to manage xApp subscriptions to RAN data and provide unified data distribution to xApps,
- Conflict Mitigation (ConMit) - allows RIC to mitigate conflicting control decisions made by xApps,

- Messaging Infrastructure - allows RIC to transfer messages between xApps and other internal endpoints within the RIC,
- Security - allows RIC to prevent malicious xApps from controlling RAN nodes and accessing network information,
- Management - allows RIC to manage the RIC and its components,
- Interface Termination - allows RIC to act as termination for relevant interfaces (A1, E2, O1, Y1),
- API Enablement - allows RIC to manage RIC Application Programming Interfaces (APIs),
- AI/ML Support - allows RIC to manage and train AI/ML models, prepare and ingest data for the models, and infer the models,
- xApp Repository Function - allows RIC to manage access to xApps and expose information about them.

Capabilities of the Near-RT RIC and its xApps are exposed as services; the RIC itself and any xApp can act both as service producer and/or service consumer. The services which can be accessed through RIC APIs are as follows [45, 55]:

- A1-related - allows service consumer to manage policies and subscriptions to enrichment information provided via A1 interface,
- E2-related - allows service consumer to manage subscriptions to information related to E2 interface, acquire guidance information, control E2 nodes, query information about E2 nodes,
- Management-related - allows service consumer to (de)register itself with Near-RT RIC,
- SDL-related - allows service consumer to store and retrieve data from/to SDL,
- API-Enablement-related - allows service consumer to manage and discover services available in the RIC,
- AI/ML-related - allows service consumer to manage, train, and infer models, and handle data for model training and inference; only provided by the Near-RT RIC.

The service-based internal representation of the Near-RT RIC is shown in Figure 2.6.

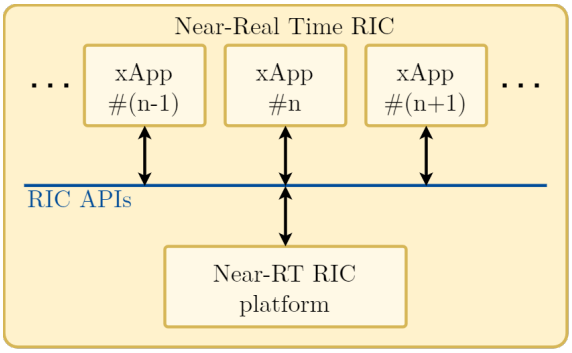

Figure 2.6: Service-based internal representation of Near-RT RIC

Source: own elaboration based on [45]

In terms of RAN control intelligence provided by xApps, the services related to the E2 interface between Near-RT RIC and RAN nodes are crucial. E2 services are realized with E2 Application Protocol (E2AP) RIC functional procedures [15, 56, 57]:

- RIC Subscription - used by E2 node and Near-RT RIC to manage event triggers in E2 nodes,
- RIC Indication - used by E2 node to carry outcome of E2 REPORT and E2 INSERT services,
- RIC Control - used by Near-RT RIC to initiate E2 CONTROL service,
- RIC Query - used by Near-RT RIC to request information related to RAN and/or UE,
- RIC Service Load Status - used by Near-RT RIC to report service level load updates,
- RIC Service Load Update - used by E2 node to report load status information for one or more E2 services,
- RIC Assistance - used by E2 node to initiate E2 ASSISTANCE service.

Other than the RIC functional procedures, the following global procedures are supported in E2AP [15, 56, 57]:

- E2 Setup - used by E2 node to exchange data needed to interoperate with Near-RT RIC on the E2 interface,
- Reset - used by E2 node and Near-RT RIC to reinitialize the state of E2 communication at the other termination of the interface; as the result of Reset procedure, any previously established RIC Subscriptions are removed and ongoing processes are gracefully terminated,
- Error Indication - used by E2 node and Near-RT RIC to report errors detected in received message,
- RIC Service Update - used by E2 node to inform Near-RT RIC about an update to application level data needed to interoperate with Near-RT RIC on the E2 interface,
- RIC Service Query - used by Near-RT RIC to discover application level data needed to interoperate with E2 node on the E2 interface,
- E2 Node Configuration Update - used by E2 node to inform Near-RT RIC about an update to its configuration,
- E2 Connection Update - used by E2 node to inform Near-RT RIC about an update to its E2 interface connections,
- E2 Removal - used by E2 node and Near-RT RIC to gracefully terminate the E2 connection.

The following E2 services are envisioned as part of O-RAN specifications [56]:

- provided by E2 nodes:
  - E2 REPORT - used by Near-RT RIC to request that E2 node sends a REPORT message to the RIC each time an event trigger related to a configured subscription happens,
  - E2 POLICY - used by Near-RT RIC to request that E2 node execute a specific policy each time an event trigger related to a configured subscription happens,
  - E2 INSERT - used by Near-RT RIC to request that E2 node sends a INSERT message to the RIC each time an event trigger related to a configured subscription happens,
  - E2 CONTROL - used by Near-RT RIC to initiate or resume a procedure in E2 node,
  - E2 QUERY - used by Near-RT RIC to retrieve information about RAN or UE from E2 node,

- provided by Near-RT RIC:
  - E2 ASSISTANCE - used by E2 node to utilize a service offered by Near-RT RIC.

Structure of an E2 packet, along with the relation between E2AP and E2SM, is shown in Figure 2.7.

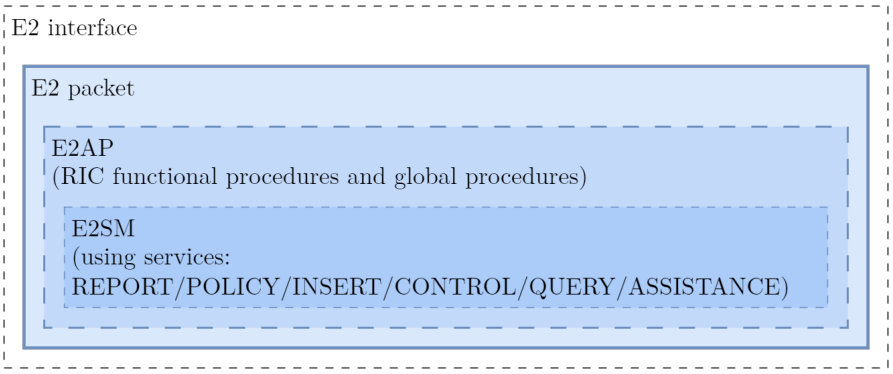

FIGURE 2.7: Structure of an E2 packet

Source: own elaboration based on [15, 50, 51, 56, 57]

There are currently five E2SMs defined in the O-RAN specifications [50]:

- RAN Control [58] - exposes information related to RAN control and UE context, allows to initiate and modify call processes and messages, allows to execute policies related to RAN control,
- Network Interface (NI) [59] - exposes information about network interfaces, allows to modify network interface message contents, allows to execute policies related to network behavior,
- Cell Configuration and Control (CCC) [60] - exposes information related to node- and cell-level configuration, allows to configure node- and cell-level parameters,
- Lower Layers Control (LLC) [61] - exposes information related to L1 and L2, allows configuration of L1 and L2 parameters,
- Key Performance Measurement (KPM) [62] - exposes KPI measurements, allows periodic reporting of measurements.

2.3.3 Use cases for Non-RT RIC and Near-RT RIC

Use cases envisioned for Non-RT RIC and Near-RT RIC as part of the O-RAN specifications include, but are not limited to [46, 49, 54, 63]:

- Traffic steering - centralized policing of the traffic steering to drive network traffic according to a specific intent of the operator,
- Quality of Experience (QoE) optimization - predictive optimization of QoE via analysis of network metrics and policing QoE targets at Near-RT RICs,
- Quality of Service (QoS)-based resource optimization - dynamic reconfiguration of RAN resources to ensure adequate QoS for prioritized traffic,
- Context-based dynamic handover management for Vehicle-to-Everything (V2X) - customization of handover sequences in V2X scenarios based on learned traffic and navigation patterns,

- RAN slice Service Level Agreement (SLA) assurance - optimization of the slicing configurations to assure SLA,
- Network Slice Subnet Instance (NSSI) resource allocation optimization - predictive optimization of NSSI resource allocation based on traffic patterns learned from performance data collected from the O-RAN NFs,
- mMIMO optimization - policing beamforming patterns to optimize mMIMO transmission for current network conditions,
- Network energy saving - cell reconfiguration and advanced sleep modes for reduced power usage,
- O-Cloud resource optimization - reconfiguration of cloud resources for optimized operation.

To provide more insight on cooperation of rApps and xApps, an example of traffic steering use case will be presented here in detail [46]. This use case is targeted for heterogeneous networks, where multiple access options are concurrently available to end users. Access may be provided by networks such as 4G LTE and 5G NR RAN working in many licensed bands, 5G NR-U RAN working in unlicensed bands, or even Wi-Fi Wireless Local Access Network (WLAN) which also works in unlicensed bands. Varied traffic with diverse throughput and QoS requirements adds another layer of heterogeneity, which management activities in the RAN need to take care of. In such networks, distributing the traffic across available access methods is a significant challenge to the network operator. Current approaches widely used in existing RAN are mainly cell-centric and passive—optimization is performed with cell-level metrics in mind instead of focusing on UE performance and the traffic steering algorithms rarely use methods of prediction of traffic patterns [46, 49].

Traffic steering using rApps and xApps aims to solve the issues with conventional traffic steering methods available in "closed" RAN. Specifically, the RAN optimization logic in Non-RT RIC uses AI/ML capabilities to enable proactive traffic management via flexible policy configuration steered from the Non-RT control loop and executed in the Near-RT control loop. The use case includes three main actors: SMO (along with the Non-RT RIC), Near-RT RIC, and RAN nodes accessible via E2 interface.

The role of the SMO in the traffic steering use case is to collect information about the network and to use that information to provide guidance to Near-RT RICs. First, SMO retrieves configuration parameters and performance metrics about RAN nodes and UEs. There are two goals of collecting this data by the SMO: to define policies guiding traffic management in Near-RT RICs and to compute A1 enrichment information based on statistical analysis of the data to assist the traffic steering process. Second, having executed the traffic management logic, SMO communicates target policies and enrichment information to the Near-RT RICs.

Near-RT RICs act mainly as enforcers of the A1 policies provided by the Non-RT RIC. Additionally, they ingest the A1 enrichment information provided by the SMO to aid the traffic management process steered via the E2 interface towards RAN nodes. The main objective of the RAN nodes in this scenario, other than to respond to E2 control performed by Near-RT RICs, is to collect and provide requested data to SMO over the O1 interface.

A specific example of the traffic steering use case in line with the responsibilities outlined above could include a UE maintaining two active connections to the RAN [46]: one carrying voice traffic with 5G QoS Identifier (5QI) equal to 1 and one carrying broadband data transmission with 5QI equal to 9. The area is covered by a network with four frequency bands, two covering larger areas in

lower frequencies (cells A and B) and two covering smaller areas in higher frequencies (cells C and D). In such scenario, based on the logic implemented within the rApp, the Non-RT RIC is likely to decide that the voice traffic should be handled by one of the low bands, while the broadband data transmission traffic by one of the high bands with potential of fallback to low band connectivity for coverage reasons.

SMO executes the traffic management logic by providing A1 policies to cells where the UE currently resides or is highly likely will soon reside in. Example of such A1 policy messages in JSON format communicated by the SMO are shown below in Listings 2.1 and 2.2 [46], for voice traffic and broadband data transmission traffic, respectively.

Listing 2.1: A1 policy configuration for voice traffic

```
{
    "policy_id": "1",
    "scope": {
      "ue_id": "1",
      "slice_id": "1",
      "qos_id": "1"
    },
    "statement": {                        // Carry C-Plane for UE on cell B as primary cell
      "cell_id_list": "B",
      "preference": "Shall",
      "primary": true
    },
    "statement": {                              // Voice U-Plane on cell B as secondary cell
      "cell_id_list ": "B",
      "preference": "Shall",
      "primary": false
    }
}
```

Listing 2.2: A1 policy configuration for broadband data transmission traffic

```
{
    "policy_id": "2",
    "scope": {
      "ue_id": "1",
      "slice_id": "1",
      "qos_id": "9"
    },
    "statement": {                // Avoid handling broadband data transmission on cells A
    and B
      "cell_id_list ": {"B", "A"},
      "preference": "Avoid",
      "primary": false
    },
    "statement": {                // Prefer broadband data transmission U-Plane on cells C
    and D as secondary cells
      "cell_id_list": {"C", "D"},
      "preference": "Prefer",
      "primary": false
    }
```

18 }

2.3.4 dApps in O-CU and O-DU

Introduction of dApps is envisioned as a significant extension to programmable RAN intelligence in O-RAN. Deployed in O-CU and O-DU, dApps operate within the RT control loop [52]. The aim of introducing these distributed apps is to reduce inference latency and communication overhead in RAN control. Taking advantage of the minimization efforts in GPUs and models, dApps are capable of utilizing inference using AI/ML techniques. Having dApps, it is possible to introduce vendor-agnostic programmability into RT control of lower layers. Specifically, dApps allow to execute control decisions at sub-frame level, where xApps and rApps working in "slower" control loops are not capable of such actions.

To enable operation of dApps, the E2 interface is extended to provide enrichment information [52]– similar to how A1 interface provides such information to Near-RT RIC and xApps. Additionally, the E3 interface is proposed between dApps and internal platform of RAN NFs (i.e., O-CU and O-DU) to enable coordination with xApps [53]. However, specification of dApps and the relevant changes to interfaces is a topic for future study in O-RAN. Official specifications published by O-RAN Alliance do not describe any details about dApps.

Despite not being officially specified by O-RAN Alliance, the existing research lists a number of use cases for dApps [52, 53, 64]:

- Physical layer security - security measures implemented on physical layer level, such as anomaly and jamming detection,
- Spectrum sensing - spectrum analysis for real-time interference mitigation via cell reconfiguration,
- Real-time scheduling - dynamic reconfiguration, acceleration or coordination of time-frequency resource scheduling process,
- Energy saving - temporal disabling of active cores in the O-DU to reduce power usage,
- AI/ML-based Channel State Information (CSI) feedback, channel estimation, and coding - compression of CSI data using ML models,
- Remote interference mitigation - mitigation of ducting interference detected during RAN operation,
- Uplink throughput optimization - adaptation of number of uplink data streams to suit to current network conditions,
- Beam management - dynamic custom beam control and pre-coding weight optimization,
- Integrated communication and sensing - extraction of sensing information requested by sensing applications,
- Traffic analysis and classification - traffic type identification for optimization of UE-to-slice allocations.

2.3.5 Landscape of RICs and apps

The presented landscape of RICs and applications in various control loops showcases the possibilities of intelligence in O-RAN. Despite the number of opportunities that these applications present to modern RAN, there are also considerable downsides.

The openness of O-RAN architecture and its interfaces is one of its key characteristics, but it also brings significant challenges. With the 3rd-party-developed applications deployed across all control loops and communicating via open interfaces, it is crucial to ensure platform-level security [3][65]. Enablement of AI/ML-powered solutions in O-RAN opens up the possibility of attacks on the ML algorithms deployed in the xApps, such as [3][66]:

- poisoning attacks by manipulating data provided to ML algorithms for model learning,
- evasion attacks by manipulating data provided to ML algorithms for model inference,
- API-based attacks by trying to extract training data by observing outputs of the ML model and discovering model's architecture and/or parameters.

Simultaneously, introduction of xApps allows the applications to implement functions for security enforcement, such anomaly detection and jamming protection [3][65]. These applications can help detecting threats before they start affecting the performance of the network.

The landscape of RICs and apps in O-RAN with interfaces relevant to RAN intelligence is summarized in Figure 2.8.

2.4 AI/ML in O-RAN

2.4.1 ML basics

Machine Learning, or ML in short, is a subfield of Artificial Intelligence focused on enabling computers to learn to solve specific problems without explicit programming [67]. Both AI and ML have seen significant developments in the recent years. Simultaneously, computing solutions using these techniques have become common even for non-technical uses. Specifically, there is a significant number of popular AI agents, such as ChatGPT and Copilot, that are capable of handling complex tasks that would not be computationally feasible using conventional AI-less logic.

The basic categorization of ML algorithms is derived from the characteristics of the training process. The following three categories, summarized visually in Figure 2.9, are considered [68, 69]:

- Supervised Learning (SL) - training data is labeled, meaning that the target result is known during the training process; covered problems include regression (predicting a quantity) and classification (predicting a label),
- Unsupervised Learning (USL) - training data is unlabeled, meaning that the target result is unknown during the training process; covered problems include clustering and principal component analysis,
- Reinforcement Learning (RL) - training is based on interaction of an RL agent with an environment, where each action of the agent leads to a reward provided by the environment; covered problems include decision-making and control tasks.

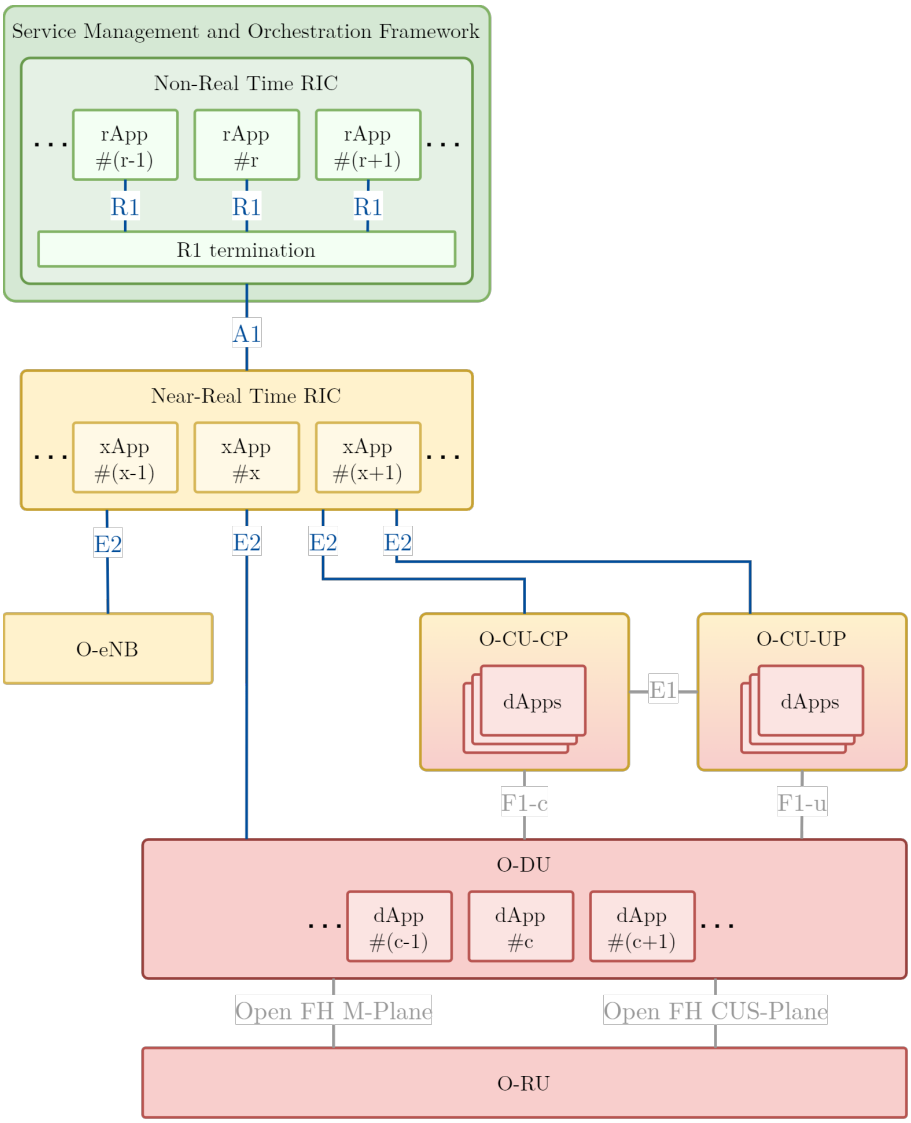

FIGURE 2.8: RICs and apps in O-RAN

Source: own elaboration based on [15, 43–45, 52]

2.4.2 ML in O-RAN

Having autonomous AI-based agents capable of computing solutions to complex problems is desirable also from the perspective of managing and optimizing operation of RAN. To take this into consideration, O-RAN specifications are developed with native support of AI/ML techniques in mind [65, 71]. RICs are the main components within the O-RAN architecture to manage ML models [44, 45, 69], but inference of ML models can happen also in the RT control loop (e.g., in O-CU, O-DU) [15, 52, 53, 71]. ML management actions performed by the RICs include collecting and preparing data, training, deploying and terminating the models, inferring the models, and communicating the data related to the ML models [68].

O-RAN specifications propose that any ML model, regardless of the used ML algorithm, is appropriately trained offline before being deployed in a live network [68]. A model deployed in the live network then can use the raw data observed during operation to further train its logic to adapt to the current network conditions.

Specific split of activities across the RICs depends on the deployment scenario, e.g., where the ML training host and model are located. O-RAN envisions a certain flexibility in locating the ML

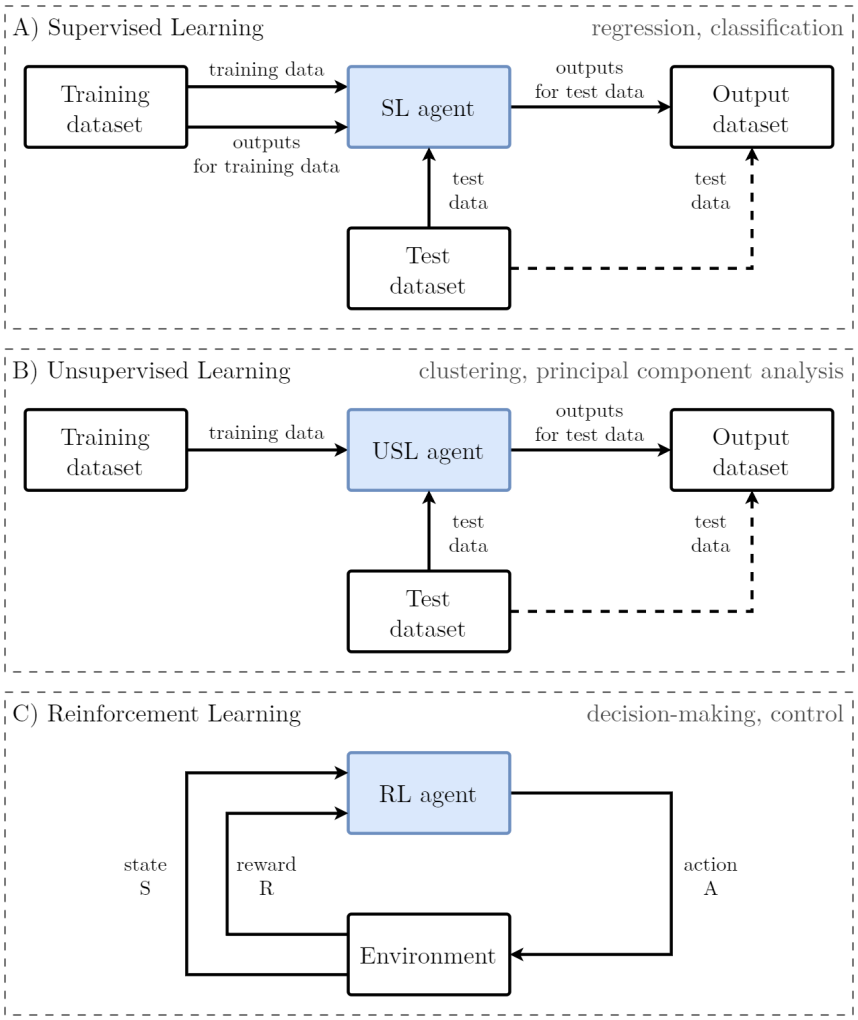

FIGURE 2.9: ML categories

Source: own elaboration based on [68, 70]

training host and model in an O-RAN deployment. However, the allowed locations are dependent on the utilized ML algorithm [68]. The supported locations of the components depending on the ML algorithm category are shown in Figure 2.10.

2.4.3 Detailed AI/ML workflow in O-RAN

Understanding the AI/ML workflow of operations in O-RAN is crucial to comprehend how ML-based applications operate and how their operation can be influenced by the MNO through management actions available in the standard. Regardless of the specific ML algorithm, the general workflow of using AI/ML solutions in O-RAN consists of several key stages, each containing a number of steps [15, 68, 71, 72]:

- AI/ML model management - performed by Non-RT RIC in the SMO or the SMO itself as the AI/ML model manager. This stage includes validation, publication and deployment of models. Validation is necessary to ensure that a model is operating as expected. A validated model can then be published and stored in the catalog of AI/ML models, then it can be deployed using the O1 interface. A model which failed validation requires retraining.
- Continuous operations - this stage includes monitoring of active ML solutions to evaluate the effects of their operation on the network. An ML solution deemed having negative impact

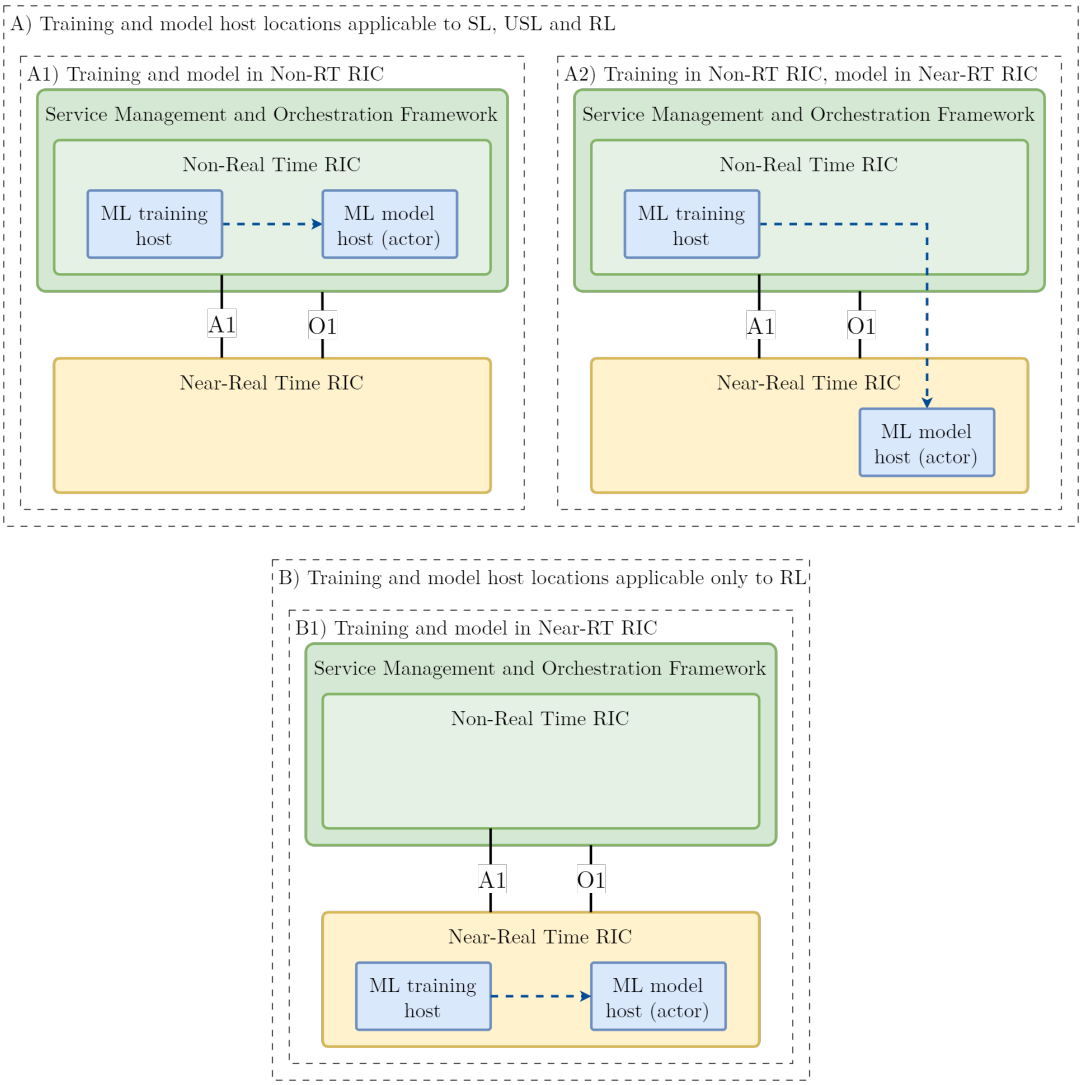

FIGURE 2.10: ML training host and model locations in O-RAN

Source: own elaboration based on [43, 68]

on the network performance and/or reliability can be retrained, reconfigured, or terminated completely.

- Data preparation - this stage includes collecting and processing raw data provided by O-RAN NFs, namely Near-RT RICs, O-CUs, O-DUs, and O-RUs. The goal of this processing is to format the raw data into the structure required by specific models, including scaling and normalization.

- Model training - this stage covers execution of the offline training of ML models, which is required for any ML solution before deployment into a live O-RAN network. Training is performed until an ML model is successfully validated by the AI/ML model manager.

- Model inference - this stage covers online inference of a trained and deployed ML model using data collected by the inference host from the live network. Based on the inference results, the inference host provides a control decision over a relevant interface. The entire inference process needs to fit within time constraints of the relevant control loop. Online inference may also include training of the model using data from live network to finetune the model.

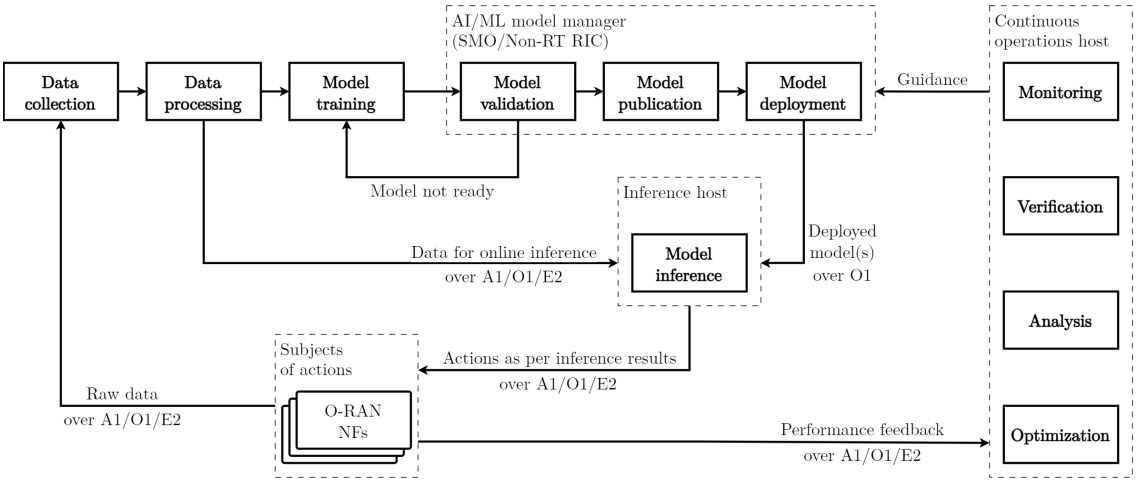

FIGURE 2.11: AI/ML workflow in O-RAN

Source: own elaboration based on [15, 68, 73]

TABLE 2.2: Deployment scenarios with split of responsibilities in AI/ML workflow in O-RAN

Source: own elaboration based on [68]

Scenario	AI/ML model management	Continuous operations	Data preparation	Model training	Model inference
Scenario 1 (offline/online)	Non-RT RIC				
Scenario 2 (offline)	SMO	Non-RT RIC			
Scenario 2 (online)	SMO	Non-RT RIC	Near-RT RIC		
Scenario 3 (offline/online)	SMO	Non-RT RIC	SMO		Non-RT RIC
Scenario 4 (offline)	Non-RT RIC		Non-RT RIC	SMO/Non-RT RIC	
Scenario 4 (online)	Near-RT RIC				
Scenario 5, for future study	Non-RT RIC				O-CU/O-DU/O-RU

A graphical representation of the AI/ML workflow in O-RAN is shown in Figure 2.11. The actual implementation of a solution may include all or only some of the steps [68].

The execution of specific AI/ML workflow stages depends on the deployment scenario and involves various O-RAN NFs. For instance, ML model inference can occur in the Near-RT RIC, O-CU, O-DU, or O-RU. Similarly, data processing may be performed within the Non-RT RIC or the SMO. Table 2.2 details these deployment scenarios and the division of responsibilities for the workflow illustrated in Figure 2.11.

2.4.4 Examples of AI/ML usage in O-RAN

The topic of utilization of AI/ML solutions to execute RAN optimization in O-RAN has already been quite widely researched. For the last several years, there were many publications regarding general AI/ML workflow in O-RAN, surveys and concept papers and the relevant fields (such as environments and datasets for easability of O-RAN application development and testing), but also articles describing specific implementations and use cases driven by various AI/ML techniques. These include solutions using ML algorithms, multi-agent, hierarchical and distributed solutions, Federated Learning (FL) and Team Learning (TL) architectures, and even agent-based solutions with Large Language Models (LLMs).

General and early contributions

Lee et al. [74] provided an early look at the AI/ML capabilities in O-RAN and proposed a dedicated workflow using open-source software such as Acumos [75] and Open Network Automation Platform (ONAP) [76] interworking with RIC software provided by the O-RAN Software Community (O-RAN SC) [77]. The implemented workflow allows to dynamically deploy xApps with ML models, with robust automated monitoring measures. Based on their experience during implementation of the workflow, the authors provide suggestions for improvements in the AI/ML workflow in O-RAN.

In an article by **Habibi et al. [78]** three deployment scenarios for AI/ML solutions hosted in the SMO are proposed. The authors focus on one of the scenarios (i.e., internal model training and deployment), where the Non-RT RIC is used to collect and process data for the ML algorithms and perform end-to-end management of the ML models. Finally, the paper provides the authors' take on the main challenges for AI/ML solutions implemented in the SMO and acting as the central data processing and ML management entity.

A survey by **Azimi et al. [79]** focuses on utilization of AI/ML in RAN slicing with the focus on resource management in 5G and beyond networks. As such, the article does not focus only on O-RAN, but still provides relevant perspective on the impact of ML solutions on RAN. The authors perform a meta analysis of existing papers on the topic, identify feasibility and applicability of the solutions proposed in them, and conclude the lessons learned and key future challenges in the field, along with some potential solutions. One of the interesting information taken from the survey is that RL using Deep Q-Learning is the most popular ML technique in the field of resource management in RAN slicing scenarios.

Reinforcement Learning

From the popular ML techniques, RL finds the most interest in research related to ML-powered RAN intelligence in O-RAN, similarly to 5G and beyond RAN in general.

Bonati et al. [80] investigated the topic by deploying an O-RAN network using open source software and Colosseum [81, 82], an advanced wireless network emulator. The authors deploy a closed-loop control solution with an ML-based xApp implemented in the Near-RT RIC, with RAN node data collection and ML model management (including offline training) implemented in the Non-RT RIC. The xApp is equipped with Deep RL (DRL) algorithms capable of selecting optimal scheduling policies for RAN slices. The experimental results show that the DRL-based xApps working in closed control loop show significant improvement in comparison to conventional fixed scheduling policies, such as 20% gain in spectral efficiency versus the best performing conventional fixed policy.

Again using Colosseum as the O-RAN testbed, **Tsampazi et al. [83]** follow up the work in [80] by benchmarking 12 xApps equipped with DRL agents targeting a subset of common optimization goals, albeit with varying design choices. The differences in the design choices include varying discount factor, action space (slicing, scheduling, or combination of slicing and scheduling), and weights used for reward computation. Furthermore, four combinations of decision-making timescales

for slicing and scheduling control were considered. The key takeaway from this work is that the design choices in xApp implementation may lead to significant changes in xApp performance despite similar optimization objectives.

Similarly, in the work by **Lee et al.** [**84**], an RL-based xApp is implemented to control O-DU parameters with the goal to maximize total cell throughput. This implementation considers training and inference to be hosted in Near-RT RIC. Performance of the proposed ML solution is over 38% higher than a reference rule-based algorithm.

Kouchaki et al. [**85**] approach the task of implementing a resource allocation xApp with different RL actor-critic techniques, namely Advantage Actor-Critic (A2C) and Proximal Policy Optimization (PPO). The target ANN is the same for both algorithms. Comparison of two methods show that both are able to converge to an optimal policy, but A2C performs better in the early stages of the simulation, while PPO converges to the optimal policy faster.

Work by Barker et al. [**86**] investigates RL-based xApp based on the actor-critic algorithm PPO for resource allocation optimization inRAN slicing. The implemented xApp allocates Physical Resource Blocks (PRBs) to various slices, optimizing the key performance metric for given slice (e.g., minimizes network delay for an Ultra-Reliable Low-Latency Communication (URLLC) slice). The proposed framework with PPO-based xApp evaluated against conventional and DRL-based resource allocation methods showed significant improvement in key performance metrics for each slice type.

In an article by **Habib et al.** [**87**], a Hierarchical RL (HRL) approach for intent-driven RAN control is investigated. HRL is a natural solution for the multi-level O-RAN RIC architecture, with higher level policies being decided in the Non-RT RIC enforced on the lower level by the Near-RT RIC. The authors propose a solution with an rApp acting as the meta-controller providing operator's optimization goals towards the Near-RT RIC, where three different xApps are acting as low level controllers for traffic steering, cell sleeping control, and beamforming optimization. The rApp observes the network state and provides guidance via A1 interface about which xApps to enable. When evaluated against a baseline scenario with no ML-based xApp coordination, the proposed solution provides a 7.5% increase in average total throughput of the network and an over 17% increase in energy efficiency.

Another paper by the same research team, i.e., **Habib et al.** [**88**], continues the topic of HRL in O-RAN optimization, this time focusing solely on the traffic steering use case. The authors propose a framework with a meta-controller rApp in Non-RT RIC providing high level traffic steering goals to a controller xApp deployed in Near-RT RIC. This proposed HRL scheme improves on the performance of conventional threshold-based and DRL-based solutions in terms of both reducing delay and increasing average total throughput of the network. Delay was reduced by

almost 59% in comparison to the conventional method and over 27% versus the DRL method; throughput was increased by over 15% and over 6%, respectively.

The article by **Orhan et al. [89]** considers the challenge of connection management in O-RAN via an xApp employing Deep Q-Learning Network (DQN) and Graph Neural Network (GNN) algorithms: network is modeled as a graph representing cell-UE relationships with channel capacities as edges between graph nodes, GNN is fed the network graph as input, while DQN is used to learn parameters of the GNN. Furthermore, the authors analyze three distinct objective functions for the connection management logic, focusing on total throughput, cell coverage, or load balancing fairness. The implemented solution showed good performance in various network and user densities, boasting up to 10% gain in total network throughput, up to 45% improvement in load balancing fairness, and up to 140% gain in network coverage in comparison to conventional greedy techniques.

Multi-agent systems

Multi-agent systems are also considered as part of research on ML-powered O-RAN intelligence. **Zhang et al. [90]** published an article, where the team learning scheme is assessed in a scenario with two xApps with the goal of power allocation and radio resource block allocation. The basic assumption for the considered team-learning scenario is that xApps are taking into account control decisions executed by other applications, allowing to optimize their logic. Such approach requires logic of all participating xApps to consider decisions of other xApps. In comparison to a scenario without cooperation between xApps, the implemented team-learning scheme can provide significant improvement in total network throughput (e.g., almost 9% for 6 Mbps traffic load) with improved resiliency to varying movements speed of the network users.

Another take on the multi-agent systems was provided by **Iturria-Rivera et al. [91]** compared three multi-agent schemes for xApps cooperating in a Near-RT RIC. The authors consider the same two xApps as Zhang et al. [90]. The following schemes were trialed: sequential (decision of xApp #2 follows decision of xApp #1), concurrent (decisions of both xApps happen at the same time) and team-learning (decisions of both xApps happen at the same time then xApps communicate their intended decision to each other), with team-learning excelling in both total throughput and energy efficiency.

Siahpoush and Shah-Mansouri [92] describe a case with two DQN-based xApps for radio resource block allocation and one xApp aiming to coordinate control decisions in conflicting scenarios. The two resource allocation xApps manage resource allocation for UEs within a region of the network; any conflicts regarding decisions for border UEs steered by both xApps are resolved by the third coordinator xApp. The proposed coordinated solution was evaluated against a centralized single steering entity and a decentralized non-coordinated scenarios. While outperforming the decentralized scenario, the proposed approach did not provide better results than the centralized one. However, the authors argue that the proposed solution was not significantly-worse-performing and provides vastly better scalability, as xApps govern only a region of the network instead of its entirety.

Rezazadeh et al. [93] in their demo work consider a scenario, where the use case for radio resource allocation is tackled using DRL and FL. The implemented testbed hosts three network slices and one double DQN agent per each slice. The agents receive rewards adequate to the their actions, learning to optimally allocate radio resources within each slice.

Federated learning

FL is another popular research topic in ML-based intelligence in O-RAN. **Abdisarabshali et al. [94]** propose a management framework dedicated for FL called Elastic FL with three additional entities for network control: Non-RT system descriptor, Near-RT FL controller, and RT FL MAC scheduler. First, the system descriptor is realized using estimator rApps used to predict FL-related parameters. Then, the FL controller handles the slicing and mobility management activities to steer resource allocation. Finally, the MAC scheduler is tasked with performing RT allocation of radio resources to UEs taking into account the various QoS indications. The proposed framework is evaluated against two baseline scenarios: one with only conventional FL methods that ignore O-RAN-specific characteristics of the network and one with a MAC scheduler which does not consider FL-specific constraints. The Elastic FL framework, by adapting to the characteristics of O-RAN, significantly improves on performance of the baseline methods providing over 85% of model accuracy throughout the considered network services.

Resolving the problem of resource allocation using FL was also considered by **Zhang et al. [95]**. Similarly to the work done on TL by the same research team [91], two xApps are considered: one for power allocation and another for slice-based resource allocation. The curious idea in this work is that the global model in the centralized FL model ingests local updates done by two various xApps aiming for different optimization goals to coordinate these applications. Other than that, the implemented FL scheme follows the usual flow. The proposed FL solution provides an over 10% total throughput performance for Enhanced Mobile Broadband (eMBB) slice and 33% delay reduction in URLLC slice.

Asad and Otoum [96] approach the topic of FL for the use case of frequency band allocation to RAN nodes in O-RAN. The training process is executed locally within the RAN nodes. The local model updates are then provided into a centralized server and aggregated there. The updated global model is then deployed into the RAN nodes. This articles does not evaluate the FL scheme in terms of performance results, but provides an analysis of its feasibility in comparison to non-FL centralized and distributed learning approaches. The authors highlight a number of benefits of FL approach to the spectrum allocation problem: reduced traffic between RAN nodes and centralized server, reduced latency due to main of the training process happening locally in the RAN nodes, enhanced network capacity by optimal spectrum allocation decisions, improvements to privacy as no raw server data is sent to centralized server, and network resilience due to the decentralized nature of the FL scheme.

The topic of Radio Access Technology (RAT) allocation is another one under analysis for feasibility of FL approach. **Erdol et al. [97]** consider the problem of allocation either 4G or 5G service to UEs based on a Federated Meta-Learning (FML) algorithm. The RAT allocation decisions can be made by RL agents in Near-RT RICs and RAN nodes using a DQN algorithm. The general FL scheme follows the typical flow: model training is done locally by the RL agents and, after a fixed number of training cycles, the model parameters are provided into a centralized server for aggregation. The newly updated global model is then propagated to the RL agents and used for the next aggregation cycle. The FML algorithm includes a meta-learning phase to the model update done by RL agents, where the agents are meant to adapt to the specific observed UE demands and move trajectories. To accomplish this, reward calculation for the RL agent consists of multiple steps, considering up to five various tasks. The authors test the implemented FML algorithm based on a simulation with five vehicles acting as UEs, moving in a 1 km by 1 km area. The proposed solution is evaluated in comparison to three methods: centralized with single DRL agent, a rule-based heuristic algorithm, and a reptile algorithm. The results show that FML provides improvements in terms of both caching and adaptation rates, showing better performance and quicker adaptability to unknown environments.

Kouchaki et al. [98] apply FL to an O-RAN network equipped with RL-enabled dApps deployed in O-DUs. Specifically, the authors consider dApps tasked with associating UEs to cells and provide them rewards based on total throughput and load balancing fairness. FL scheme follows the usual flow. The FL-based solution is compared to a centralized solution with a single xApp in the Near-RT control loop using the same logic for every O-DU and its associated UEs. Results of a network simulation show that the proposed solution is able to converge to an optimal policy and that it requires significantly less operation time in comparison to the centralized approach, proving better delay characteristics and scalability of the FL method.

Others

With the advent of LLMs in the recent years, it is not surprising that they are being evaluated also in context of O-RAN intelligence. **Wu et al. [99, 100]** propose a framework of LLM-powered agents for radio resource allocation and resilience optimization in O-RAN slices. OpenAI's GPT LLM models are used in this work. The AI agents are deployed as xApps in Near-RT RIC and steer the behavior of RAN nodes via E2 interface. Agents are operated through fine-tuned meta-prompts fed with descriptions of current network conditions and they output the resource allocation parameters. The goal of the agents is to distribute resources fairly and to reliably accommodate QoS-driven demands; this target is captured in the reward function for the agents. Memory in form of combinations of past actions, states, and rewards is provided to agents in the meta-prompts. The proposed LLM-powered solution was evaluated against three approaches: random allocation, equal allocation, and demand-proportional allocation. Being capable of adaptation to current network condition, the proposed approach showed significant improvement over the baseline algorithms in key metrics where the conventional approaches underperformed, like system utility and reliability. Simultaneously, it allowed to achieve adequate bitrate for both UEs considered in the simulation scenario.

Deviating from the most common RL-based approaches, **Qazzaz et al. [101]** provide a look at a SL random forest classifier algorithm employed for the resource allocation task in the Near-RT control loop. In their work, the authors propose a resource allocation xApp, which optimizes the network in two ways: allocates available radio resources to network slices and chooses one of multiple possible resource allocation policies for each slice. The implemented xApp, trained using the random forest classifier algorithm, showed high performance and adaptability in the simulated scenarios. These results are in contrast to fixed policies, which lead to many outages. Furthermore, the results showed that usage of random forest classifier enabled significantly shorter training time in comparison to support vector classifier and gradient boosting classifier algorithms.

It is important to also mention works which enable more accessible research of AI/ML techniques in O-RAN optimization. Significant efforts in this area were made by **Bonati et al. [102, 103]** to implement OpenRAN Gym—an open and publicly available toolbox for development of AI/ML-enabled solution for O-RAN. This research platform is capable of integrating with the Colosseum network emulator [81, 82] to test the developed solutions in a softwarized RAN. The authors describe in detail the framework of the platform along with the relevant procedures and demonstrate how OpenRAN Gym can act as the enabler for development of applications directly applicable to various real-world platforms.

Table 2.3 synthesizes the key AI/ML-based solutions for O-RAN analyzed in this section, organizing them by the primary ML technique and application domain.

Table 2.3: Summary of AI/ML-based mechanisms in O-RAN literature

Author	**Ref.**	**Algorithm type**	**Use Case / Problem Addressed**
Lee et al.	[74]	General Workflow	Automated deployment/monitoring of ML xApps
Habibi et al.	[78]	SMO-based Learning	Internal model training/management in Non-RT RIC
Bonati et al.	[80]	Deep RL (DRL)	Slicing scheduling optimization
Tsampazi et al.	[83]	DRL Benchmarking	Comparison of xApp design choices
Lee et al.	[84]	RL	O-DU parameter control for throughput maximization
Kouchaki et al.	[85]	Actor-Critic (A2C, PPO)	Resource allocation convergence comparison
Barker et al.	[86]	PPO	Resource allocation in RAN slicing
Habib et al.	[87]	Hierarchical RL (HRL)	Traffic steering, cell sleeping, beamforming
Habib et al.	[88]	Hierarchical RL (HRL)	Traffic steering (Meta-controller rApp + xApp)
Orhan et al.	[89]	DQN + GNN	Connection management (Throughput/-Coverage/Fairness)
Zhang et al.	[90]	Team Learning	Power and radio resource block allocation
Iturria-Rivera et al.	[91]	Multi-agent Schemes	Evaluation of sequential vs. concurrent vs. team learning

Continued on next page

Author	Ref.	Algorithm type	Use case / problem addressed
Siahpoush & Shah-Mansouri	[92]	Distributed DRL	Regional resource allocation with coordinator xApp
Rezazadeh et al.	[93]	Double DQN + FL	Resource allocation in slices
Abdisarabshali et al.	[94]	Elastic FL	Slicing and mobility management
Zhang et al.	[95]	Federated DRL	Power and slice-based resource allocation
Asad & Otoum	[96]	Federated Learning	Frequency band/spectrum allocation
Erdol et al.	[97]	Federated Meta-Learning	RAT allocation (4G vs 5G) for specific UE trajectories
Kouchaki et al.	[98]	FL with dApps	UE-to-cell association (Load balancing)
Wu et al.	[99, 100]	LLM (Generative AI)	Agent-based resource allocation and resilience
Qazzaz et al.	[101]	Random Forest (Supervised)	Resource allocation policy selection
Bonati et al.	[102]	Research Platform	OpenRAN Gym toolbox for AI/ML development

2.5 Future of RAN intelligence with AI-RAN

AI-RAN is a new initiative promoting closer integration of AI technology into RAN hardware and software. The AI-RAN Alliance was founded in 2024 by a group comprising of multiple industry and academic research leaders, including RAN vendors, MNOs, and universities. There are no standard specifications planned to be released by the group, as it does not consider itself a standard development organization similar to O-RAN Alliance or 3GPP. So far, at the time of writing, there are also no blueprints and specific guidelines released by the group, but the concept of AI-RAN and the vision behind the movement are presented in a white paper [104].

The main mission of AI-RAN is to embrace AI in various fields in RAN to improve network performance, provide new revenue streams for the participating parties, and drive the RAN technology development towards 6G and beyond networks. The AI-RAN Alliance identifies three key development areas, with each having its own dedicated Working Group (WG) [104, 105]:

- AI-for-RAN (WG1) - focused on using AI for RAN optimization for increased reliability, improvement of spectral and operational efficiency,
- AI-and-RAN (WG2) - focused on establishing a shared infrastructure for efficient concurrent hosting of RAN and AI workloads, unifying the hardware layer for the RAN and non-RAN applications,
- AI-on-RAN (WG3) - focused on enabling new services for RAN, with AI-powered applications running at the network edge.

Roadmaps for work in each WG within the AI-RAN Alliance are due to be released soon [104], but no specific dates are available so far.

In the future, once reference and guideline materials are released by the WGs in AI-RAN Alliance, the RAN following specifications by O-RAN Alliance and 3GPP may be implemented also in line with these recommendations.

A white paper on AI-RAN prepared by SoftBank [106] showcases more in-depth look into commercial reasons to pursue AI-RAN from the perspective of an MNO. The paper highlights the transformation of RAN from being considered a cost center to it being a generator for profit. On one hand, this would be accomplished by reducing the operational cost of the RAN with AI-RAN. On the other hand, it is envisioned that AI-RAN will enable new revenue streams via AI computing resources integrated into the data centers. These robust computing resources can then be flexibly allocated between AI and RAN applications, depending on the current needs.

Chapter 3

Conflict management in O-RAN

3.1 Conflict landscape

Although O-RAN's flexibility and novel logical components for intelligent RAN allow for robust RAN optimization, these features also create unique challenges in controlling the network. Control conflicts are posed to happen as multiple agents can simultaneously decide on RAN operation parameters for the same control targets [14]. These conflicts can happen at many levels in the O-RAN architecture, leading to various undesired effects, such as network instability and wasteful resource assignment [107]. This section details the conflict landscape as identified and categorized in the author's foundational work [2], which established the taxonomy for horizontal and vertical conflicts in O-RAN prior to the release of standardized technical reports on the topic.

Within RICs, conflicts can occur between xApps in a Near-RT RIC and between rApps in the Non-RT RIC when control activities of applications contradict each other. Furthermore, as addition of dApps is considered in O-RAN, conflicts may also arise between dApps deployed in a O-CU or O-DU. Other than that, Near-RT RICs may also conflict each other, specifically if they control network areas with RAN nodes located in close proximity. These conflicts between equivalent components (RICs, applications) can be referred to as horizontal conflicts. Control conflicts can also happen between components on different levels of the O-RAN architecture, i.e., between control loops. An example of such a type of conflict is when an xApp acts against a policy provided via the A1 interface, or contradicts a control decision provided by the Non-RT RIC. These conflicts can be named vertical conflicts. All considered areas of potential control conflict within the O-RAN architecture, first described by the author in [2] along with the categorization of horizontal and vertical conflicts, are highlighted in red in Figure 3.1.

While conflicts can happen on multiple levels of the architecture [1], O-RAN technical specifications focus mainly on the intra-Near-RT RIC conflicts between xApps. The ConMit technical report published by O-RAN Alliance distinguishes three types of control conflicts that can happen between xApps [17, 45], namely direct, indirect, and implicit. The first of these are direct conflicts, which concern contradicting decisions that happen one after the other and affect the same set of configuration parameters. For instance, a direct conflict arises if xApp #1 configures the Cell Individual Offset (CIO) of a target cell to a specific value, and immediately afterwards, xApp #2 reconfigures the same CIO parameter to a different value. In this particular scenario, the result of the conflicting actions will be that only the decision of xApp #2 will have an effect. Another clear example of a direct conflict occurs when two distinct energy saving xApps attempt to control the administrative state of the same O-RU. If xApp #1 sends a command to deactivate the O-RU to save power, while xApp #2 sends a command to activate the same O-RU to improve coverage, the

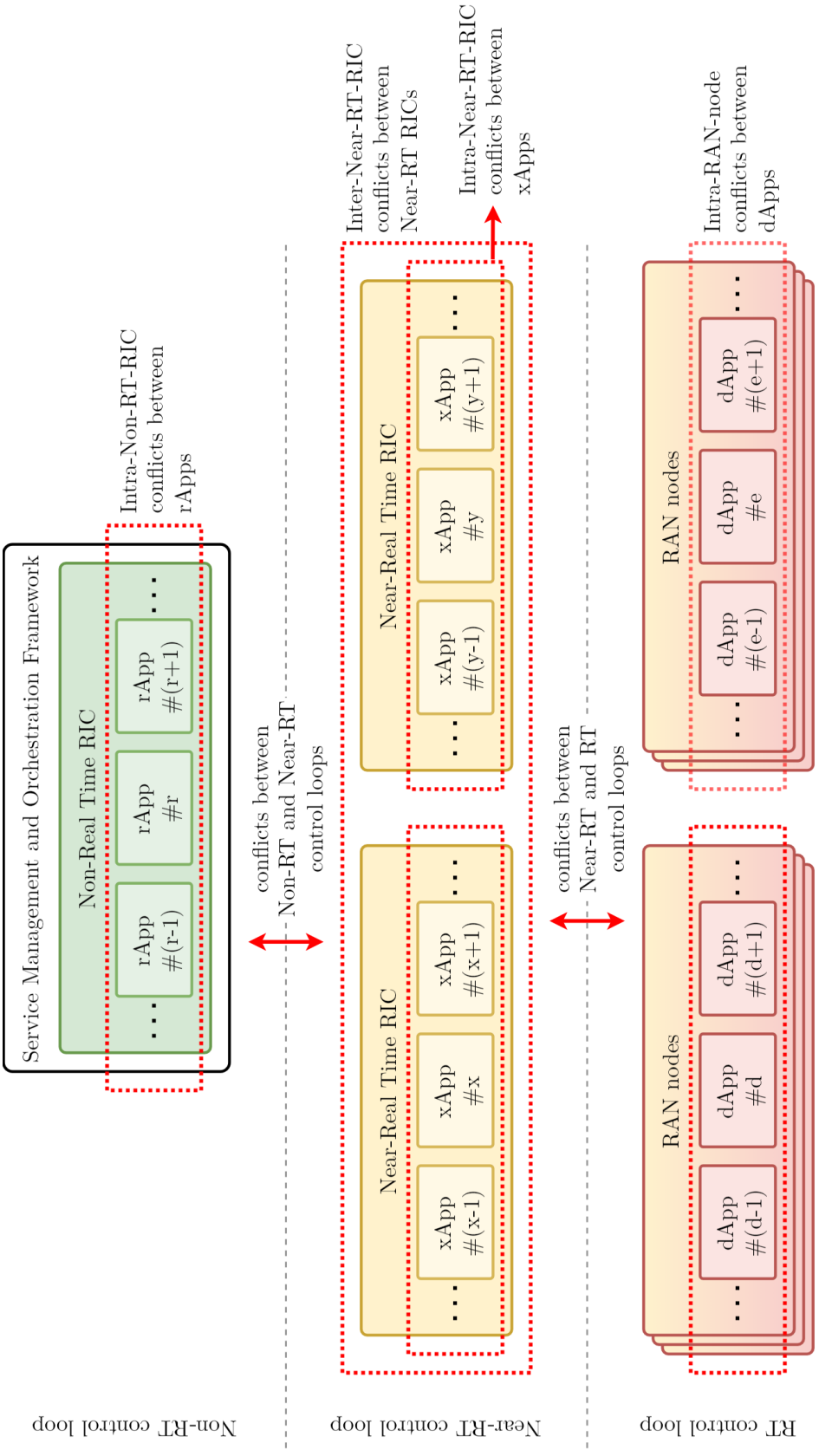

FIGURE 3.1: Conflict landscape in O-RAN

Source: own elaboration

final state of the hardware depends entirely on the race condition of message arrival, leading to network instability.

Another type of conflict in the Near-RT RIC is the indirect conflict. It refers to a situation where control decisions contradict each other when modifying parameters that influence the same areas of RAN operation. For example, indirect conflicts can happen between xApps modifying parameters that influence handover boundaries, such as when xApp #1 controls an antenna's electrical tilt and xApp #2 changes the cell's CIO. This may lead to situations where the effective handover boundary fluctuates heavily due to xApps reacting to each other's contradicting decisions.

The implicit conflict considers situations in which conflicting influences of network control decisions are difficult to observe and determine. Implicit conflicts are expected when many xApps simultaneously optimize the RAN operation with separate goals, controlling different network parameters. They can appear in a situation where xApp #1 aims to maximize QoS for a group of users while xApp #2 attempts to minimize the number of handovers between neighboring cells; decisions made by these two xApps may interfere in a non-obvious manner, disrupting each other's effect on the network.

Conflicts may also happen between Near-RT RICs that control RAN nodes in close proximity (i.e., the inter-Near-RT RIC conflicts) [1]. In such cases, the conflicting Near-RT RICs provide contradicting RAN control decisions, e.g., triggering a "ping-pong" handover of a user between cells managed by both Near-RT RICs. Such behavior of the network leads to inefficient utilization of radio resources.

As defined earlier, vertical conflicts happen in O-RAN architecture between control loops [2]. Specifically, a vertical conflict may happen between Non-RT RIC and Near-RT RIC when an xApp deployed in the Near-RT RIC acts against an A1 policy provided by the Non-RT RIC. According to the A1 interface type definitions [108], such non-adherence can be formally recognized and communicated by the Near-RT RIC through the `EnforcementReasonType` enumeration in a notification regarding policy enforcement status change. This enumeration defines three distinct cases for why a policy may be ignored: `SCOPE_NOT_APPLICABLE`, indicating the control target of the policy (e.g., an UE) are no longer valid (e.g., because the UE disconnected from the network); `STATEMENT_NOT_APPLICABLE`, indicating the specific policy target cannot be enforced (e.g., due to lack of resources); and `OTHER_REASON`, which serves as a catch-all for internal errors or ConMit overrides. While the first two reasons provide explicit feedback, the usage of `OTHER_REASON` for undefined scenarios prevents the Non-RT RIC from diagnosing the root cause of the non-adherence, effectively creating an opaque vertical conflict. Furthermore, there can be "silent" conflict scenarios where the Near-RT RIC reports a policy status `ENFORCED` despite it not being consistent with its actual control decisions; this may occur when internal safety guards of the xApp prevent specific actions to preserve stability, or when semantic mismatches cause an xApp to interpret the policy intent differently than expected in SMO, leaving the Non-RT RIC unaware that its guidance is being misunderstood or completely disregarded. This kind of conflict, where policy guidance from Non-RT control loop is ignored, leads to unpredictability of network operation and, in turn, inability of the Non-RT RIC to coherently correlate its control activities to the resulting network behavior.

Vertical conflicts can also arise between Near-RT and RT control loops, i.e., when the control actions made by Near-RT RICs and RAN nodes are not aligned. For example, in an O-RAN deployment running an xApp for QoS-based resource optimization and a dApp for real-time scheduling, the actions of the dApp may inadvertently counteract the xApp's optimization efforts by limiting the radio resources allocated to a data stream carrying high-QoS traffic.

O-RAN specifications envision that the mitigation of conflicts is realized by ConMit logical components within Non-RT and Near-RT RICs. In both cases, the component is responsible to

detect and resolve conflicts on the level of the control loop of the RIC that occur during the network's operation. Although the O-RAN architecture sections out the ConMit logic only in the RICs and it is the main focus of this work, some ConMit capabilities could be also realized on the SMO level (and outside of the Non-RT RIC).

Control conflicts happen in the context of RAN intelligence logic executed as part of O-RAN operation. The RAN control data flow in O-RAN architecture as per the current state of O-RAN Alliance specifications is based on interfaces as described in Chapter 2. Starting from the "slowest" Non-RT control loop, the Non-RT RIC hosts rApps, which execute logic for Non-RT RAN control via A1 policies management and Enrichment Information (EI). Simultaneously, policy modifications made by rApps are governed by the ConMit functionality of the Non-RT RIC, ensuring that conflicting policy management activities are handled by rejecting or suspending some of them. The policy updates and EI are provided into Near-RT RICs via the A1 interface, along with actions related to ML model management. On the other end of the A1 interface is the Near-RT RIC with its xApps, which send requests for EI and provide results of inference of ML models. Within the Near-RT RIC, xApps execute Near-RT RAN control logic and may use E2 guidance service to receive information from the RIC to resolve potential control conflicts pre-action. Applications deployed in the Near-RT RIC should utilize A1 policies and EI to execute RAN control logic in line with the MNO's current intended optimization goals. Then, they perform the RAN control via E2 messages sent to specific O-CUs and O-DUs. The efficiency of RAN control activities can be evaluated within the RICs based on performance management data reports provided by the RAN nodes. Other than the architecture components specified by documentation provided by the O-RAN Alliance, research proposes introduction of dApps into O-CUs and O-DUs RAN nodes–there are no proposals to include ConMit functionality into the RAN nodes themselves. The flow of data for ConMit as envisioned in O-RAN specifications, including introduction of dApps, is shown in Figure 3.2.

An example context of basic ConMit operation in O-RAN's Near-RT RIC is shown in Figures 3.3 and 3.4 to illustrate the operation of the ConMit component. The first figure presents a general view of a ConMit scenario, where an xApp provides a control decision for a control target (step 1), then another xApp does the same (step 2), clashing with the previous decision. The conflict is then detected (step 2) and resolved (step 3) by the ConMit component in the Near-RT RIC. In the presented case, the conflict was resolved by rejecting one of the conflicting control decisions, while retaining the other one without modification.

Figure 3.4 shows the same situation, but in a more specific setting. In this case, the two xApps both change parameters which impact handover boundaries of a cell (xApp #1 in step 1, xApp #2 in step 2). As such, these xApps are in indirect conflict with each other. When both provide control decisions to reconfigure the same 5G cell, the ConMit component detects the indirect conflict (step 2) and resolves it by rejecting the decision of xApp #2 to increase the electrical tilt of the cell (step 3).

3.2 Challenges for conflict mitigation

Table 3.1 provides an overview of the key challenges identified in the domain of conflict mitigation, which are analyzed in detail in the following subsections.

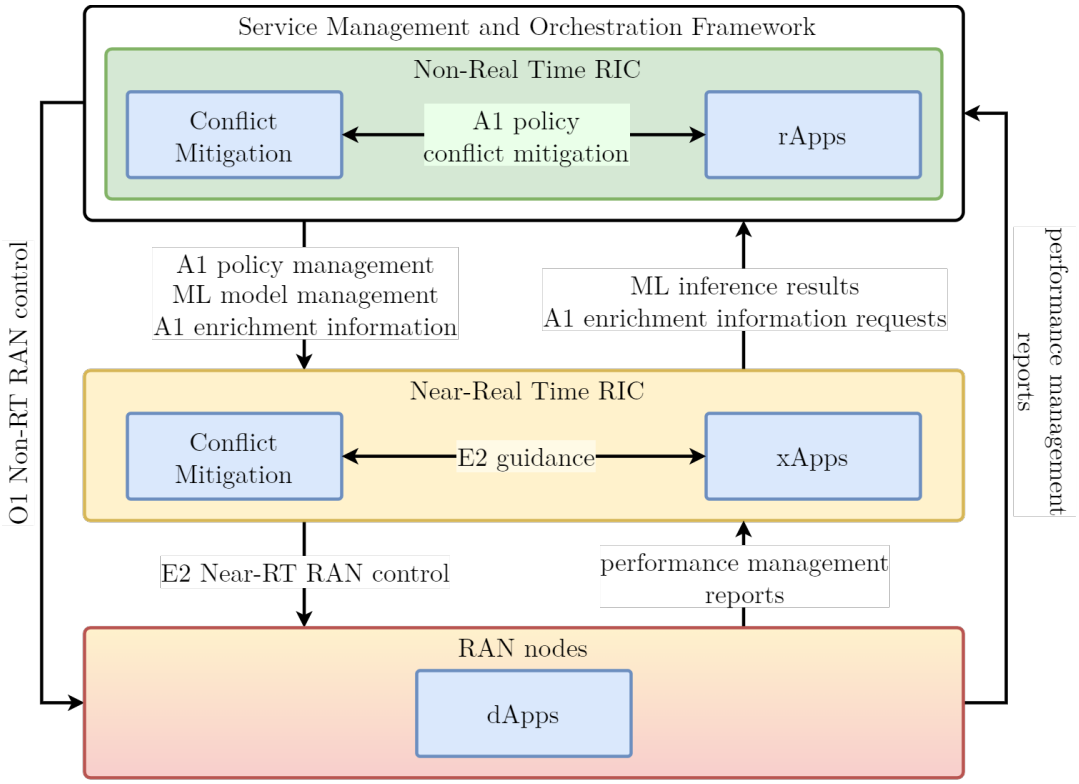

FIGURE 3.2: Conflict mitigation data flow in O-RAN architecture

Source: own elaboration based on [43, 52, 56, 63]

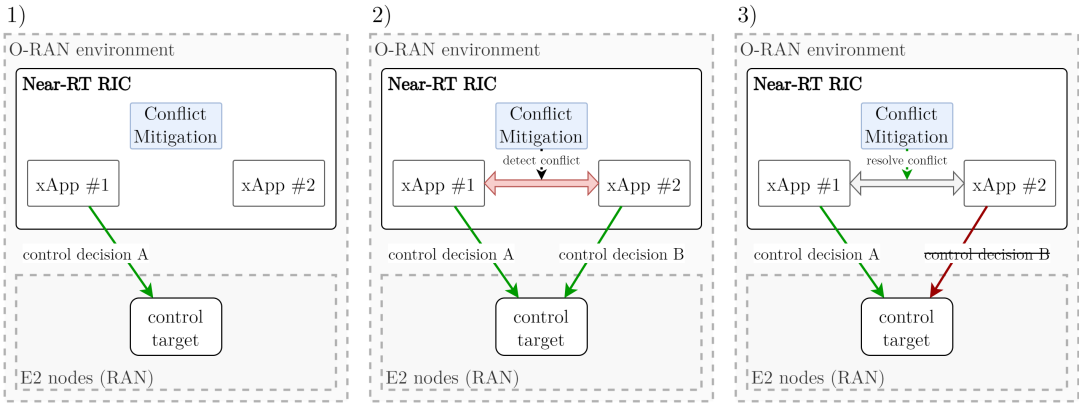

FIGURE 3.3: General context of conflict mitigation in Near-RT RIC

Source: own elaboration

3.2.1 Reliability of conflict detection mechanisms

Challenge description

The complexity of the O-RAN intelligence mechanisms makes it challenging to ensure reliable conflict detection schemes, particularly in the context of inter-Near-RT-RIC conflicts and conflicts between Near-RT and Non-RT control loops. This is due to the lack of dedicated interfaces between RICs to communicate their decisions to each other: Near-RT and Non-RT RICs utilize xApps and rApps to facilitate intelligent RAN control and these components are not inherently aware of each other's decisions, leading to potential conflicts between multiple agents providing RAN control decisions. In this situation, the lack of coordination and standardization between the various components responsible for providing RAN control decisions can result in significant complexity of

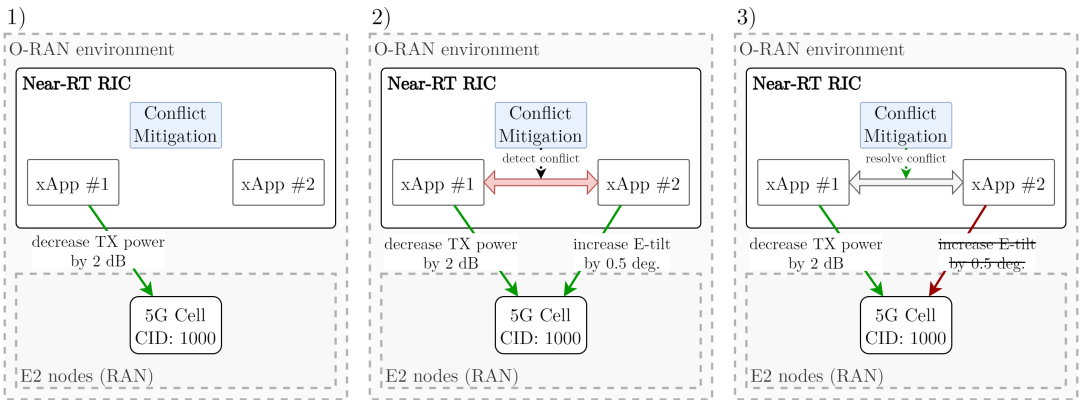

FIGURE 3.4: Example context of conflict mitigation in Near-RT RIC

Source: own elaboration

TABLE 3.1: Overview of challenges for Conflict Mitigation in O-RAN

Challenge Area	**Core Issue**
Reliability of conflict detection mechanisms	Complexity of dependencies and lack of coordination between control agents (especially across different RICs and control loops) hinders reliable conflict detection.
Maintenance of the ConMit configuration in an evolving network	Manual updates to ConMit configuration are error-prone and not scalable in networks with frequent addition, removal, or modification of applications.
Optimal conflict resolution logic	Finding a solution that optimizes multiple conflicting objectives simultaneously under real-time constraints is computationally difficult and often requires trade-offs.
Methodologies for testing and evaluation of ConMit methods	Lack of established standards, uniform testing frameworks, and objective metrics makes it difficult to evaluate and compare ConMit methods.
Limited observability of network control agents	The distributed nature of O-RAN and the opacity of internal decision-making in control agents ("black boxes") hinder conflict identification and root cause analysis.

dependencies between the O-RAN radio control agents, making it difficult to manage and mitigate conflicts effectively. Regardless of the conflicting agents, reliable conflict detection is critical for ConMit methods to take action and resolve the conflicts.

The general idea of this challenge is presented in Figure 3.5: depending on the deployed Conflict Detection (CD) method and its configuration, the ConMit functionality in the RIC may be able to detect some, but not all, conflicts happening in the O-RAN deployment. The detected conflicts are indicated with green dotted circle, while the undetected ones are marked with a red outline. Specifically, the issue of unreliable detection of conflicts is most likely to consider implicit conflicts, which are the most complex to detect.

Guidelines for addressing the challenge

Addressing the challenge of reliable conflict detection mechanisms in O-RAN requires the development of advanced algorithms and analytics that can effectively monitor and detect conflicts in runtime, as well as the deployment of robust data sharing and communication mechanisms between various logical components within the O-RAN architecture. This may involve the use of AI and ML techniques to identify and mitigate conflicts dynamically, as well as the development of standardized interfaces and protocols for exchanging network state information between different network control agents.

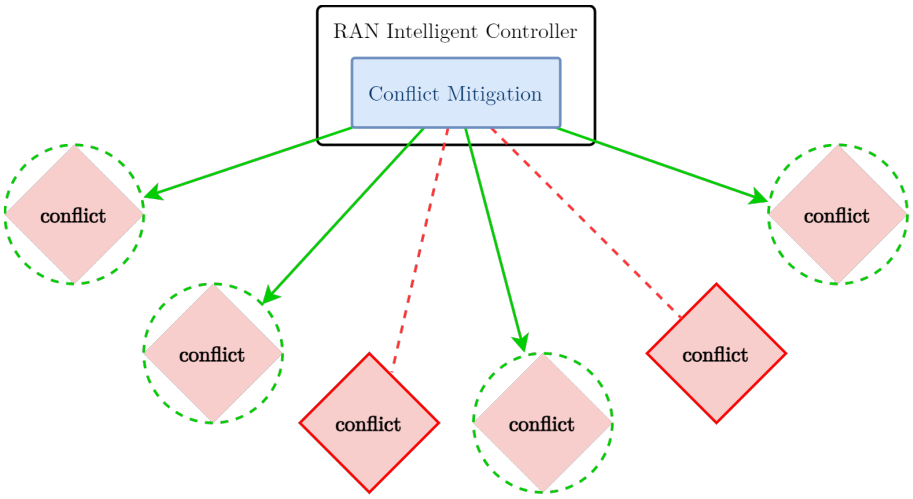

FIGURE 3.5: Challenge for conflict mitigation: reliability of conflict detection mechanisms

Source: own elaboration

3.2.2 Maintenance of the conflict management configuration in an evolving network

Challenge description

The challenge of maintaining the conflict management configuration in an evolving O-RAN network arises due to the addition, removal, or modification of applications (rApps, xApps, dApps) deployed in the environment. This maintenance is necessary to ensure the continued reliable and efficient operation of the network along with the rising complexity of the RAN intelligence measured in the network. However, manual updates of the conflict management configuration can be time-consuming and error-prone, especially in large and complex networks. Additionally, the conflict management configuration must consider not only conflicts between the RAN control agents but also between the ConMit mechanisms themselves – the implemented solutions must not interfere with each other and manage the conflicts in a consistent and predictable manner. Therefore, it is essential to develop efficient and automated methods for maintaining and updating the conflict management configuration.

The general idea of this challenge is presented in Figure 3.6: a relatively simple configuration of O-RAN RICs may be easy to manage from ConMit perspective, but after years of evolution the same network may include many more conflicting control agents, which need to be managed.

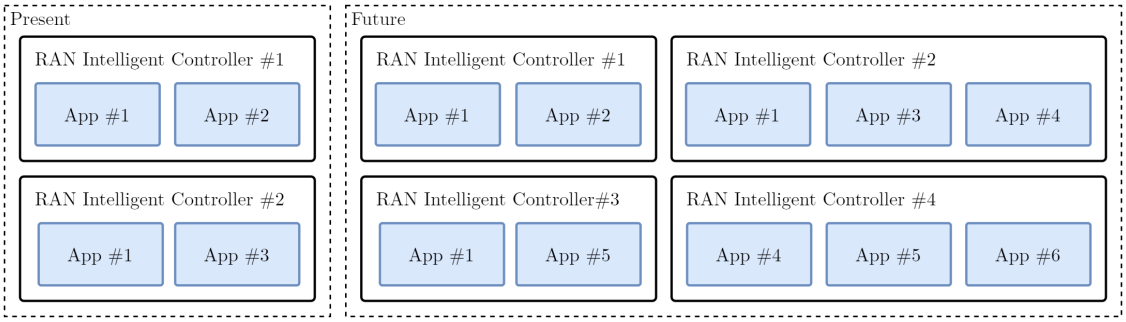

FIGURE 3.6: Challenge for conflict mitigation: maintenance of the conflict management configuration in an evolving network

Source: own elaboration

Guidelines for addressing the challenge

To support efficient maintenance of the ConMit configuration, it is necessary to invest in training and support for network operators to ensure that they have the knowledge and skills necessary to effectively manage conflicts in O-RAN deployments. In this context, it might be necessary to adapt functionality of Operations Support System (OSS) and SMO for O-RAN network management to incorporate conflict management capabilities. As such, these systems should provide the MNOs with detailed insight about the active network control agents, potential conflict areas, guidelines to preemptively resolve the conflicts, detect any potential changes in control agent configuration that may cause conflicts, etc.

3.2.3 Optimal conflict resolution logic

Challenge description

Ideally, the conflict resolution logic must be able to analyze the potential solutions to a conflict and select the best option based on several factors, including network performance, service availability, and user experience. Additionally, to ensure that conflicts are resolved optimally, the conflict resolution logic must be able to adapt to changes in the network environment, including the addition or removal of network components. This challenge requires the development of sophisticated algorithms and decision-making processes that can analyze complex scenarios and make informed decisions in the shortest possible time frame.

Finding and applying optimal conflict resolution logic can be particularly challenging in complex O-RAN radio control scenarios where there are multiple conflicting objectives, such as minimizing interference, optimizing resource allocation, and ensuring quality of service. In such scenarios, finding a solution that optimizes all objectives simultaneously may not be feasible. As such, some trade-offs may need to be made between conflicting objectives. The chosen trade-off should be aligned with the current policies applied by the MNO.

Additionally, the optimal conflict resolution logic may vary depending on the network conditions, such as the number of active users, their location, and the type of services they are using. This makes it difficult to develop a single optimal logic that works in all scenarios. Finally, the real-time nature of the radio access network also adds to the complexity, as conflicts may need to be resolved quickly to avoid service disruptions, while ensuring that the resolution does not introduce new conflicts or negatively impact other network objectives.

The general idea of this challenge is presented in Figure 3.7: a conflict appears in the network (step 1), the ConMit functionality of the RIC reacts to it in the managed network by detecting it (step 2) and applying some resolution activities (step 3) to "defuse" the conflict; the conflict is resolved (step 4).

Guidelines for addressing the challenge

A promising approach to solving the issue of applying optimal conflict resolution logic in O-RAN radio control is to develop a systematic and automated method, potentially using ML and AI techniques, that can identify the root cause of the conflict and select the best resolution strategy based on a large set of factors. These could involve network parameters, traffic patterns, applied policies, and MNO's priorities.

Another approach could be to use game theory to model and analyze the interactions between the different RAN control agents and develop optimal conflict resolution strategies based on the

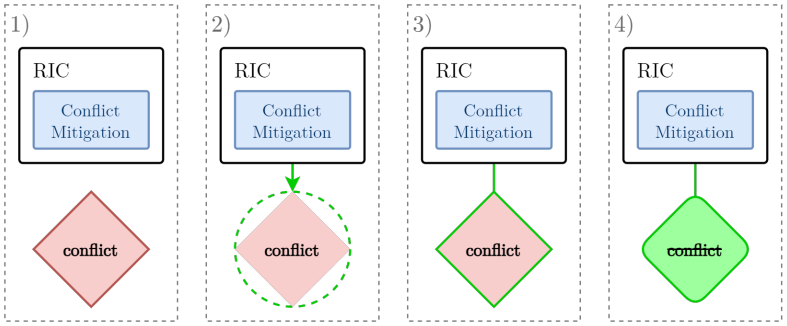

FIGURE 3.7: Challenge for conflict mitigation: optimal conflict resolution logic

Source: own elaboration

resulting game-theoretical models [18, 109]. This method would require a deep understanding of the incentives and motivations of the different RAN control agents and the ability to design mechanisms that align their interests with the overall goals of the MNO.

3.2.4 Methodologies for testing and evaluation of conflict mitigation methods

Challenge description

Defining efficient methodologies for testing and evaluating of ConMit methods in O-RAN is a challenge primarily related to the lack of established standards and guidelines for testing and evaluating the performance of these methods. The complexity and heterogeneity of the O-RAN architecture make it difficult to define a uniform testing framework that can account for the unique combinations of different RAN control agents and the various interactions between them. Furthermore, the testing methodology should account for the dynamic nature of the network and the changing traffic patterns, which can affect the effectiveness of ConMit methods.

In addition, the lack of objective metrics for evaluating the performance of conflict detection and resolution methods is also a major challenge. Without clear and quantitative metrics, it is difficult to compare the effectiveness of different ConMit methods or to identify areas where improvements can be made. Another challenge is the lack of access to realistic testbeds that can provide a representative environment for testing and evaluating ConMit methods.

The general idea of this challenge is presented in Figure 3.8: the ConMit functionality of the RIC is aware only of the conflicts which it can detect. Consequently, without an appropriate methodology to evaluate a ConMit mechanism, a wrong conclusion may be made about its efficiency. In the figure, a ConMit component might successfully resolve all detected conflicts, yet fail to identify others entirely. Consequently, the component overestimates its own efficiency by ignoring the undetected issues.

Guidelines for addressing the challenge

A potential way to approach this challenge would be to develop a systematic testing and evaluation framework that considers multiple scenarios of conflicting RAN control actions. This framework should take into account the dynamic nature of the network and be able to adapt to changing traffic patterns. It should also include a range of test scenarios that cover different network topologies, traffic loads, and service requirements. To ensure the reliability and reproducibility of the results, the testing methodology should be standardized and validated using a range of testbeds that provide a realistic representation of the O-RAN network. The development of such

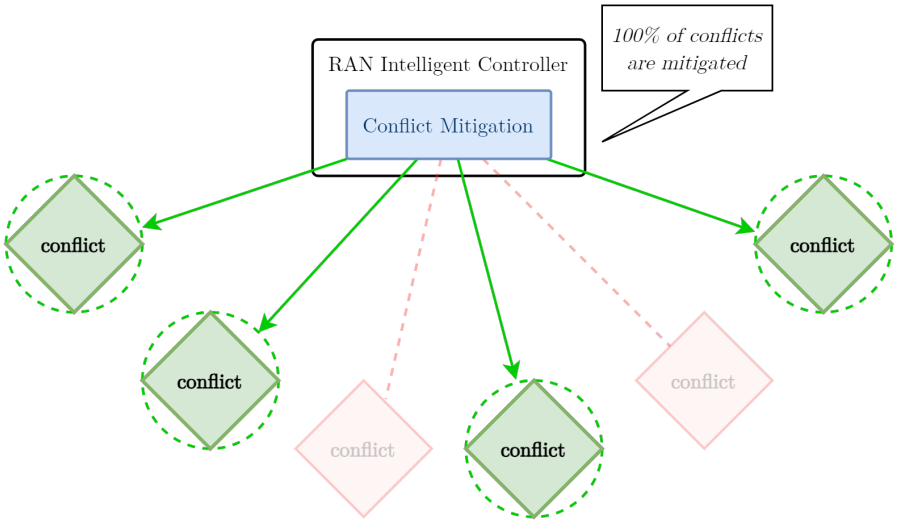

FIGURE 3.8: Challenge for conflict mitigation: methodologies for testing and evaluation of conflict mitigation methods

Source: own elaboration

a systematic testing and evaluation framework requires close collaboration between researchers, vendors, and mobile network operators to ensure the developed methodologies align with the requirements of all relevant parties.

One possible solution to the challenge of the lack of objective metrics for O-RAN ConMit is to define a set of standard metrics that can be used to evaluate the effectiveness of different ConMit methods. These metrics could include factors such as the rate of false positives and false negatives in conflict detection, the time required to detect and resolve conflicts, and the impact of conflict resolution on network performance and user experience. To ensure that the metrics are meaningful and useful, they should be developed in collaboration with industry stakeholders, including mobile network operators, equipment vendors, and research organizations. Moreover, the metrics should be designed to be flexible and adaptable to new types of applications and network control activities.

3.2.5 Limited observability of network control agents

Challenge description

The limited observability of network control agents is another significant challenge in O-RAN ConMit. Network control agents are responsible for making decisions related to network configuration and optimization, but their internal operations and decision-making processes are by default opaque to other agents in the system. Given the distributed nature of O-RAN architecture, it can be challenging to gather sufficient information about the network to effectively detect and resolve conflicts. This lack of visibility can make it difficult to identify conflicts and understand their root causes.

In addition, some network control agents may not be designed to report certain information, which limits the observability of the network. For example, a Near-RT RIC may not be designed to report specific details about its decision-making process or the state of its internal variables, which can make it difficult to determine if conflicts are arising due to a specific decision or variable. Overall, limited observability can hinder the ability to diagnose and resolve conflicts, leading to inefficient and/or unreliable ConMit.

The general idea of this challenge is presented in Figure 3.9: the ConMit functionality of the highlighted RIC is not aware of decision-making processes running in the other Near-RT RICs.

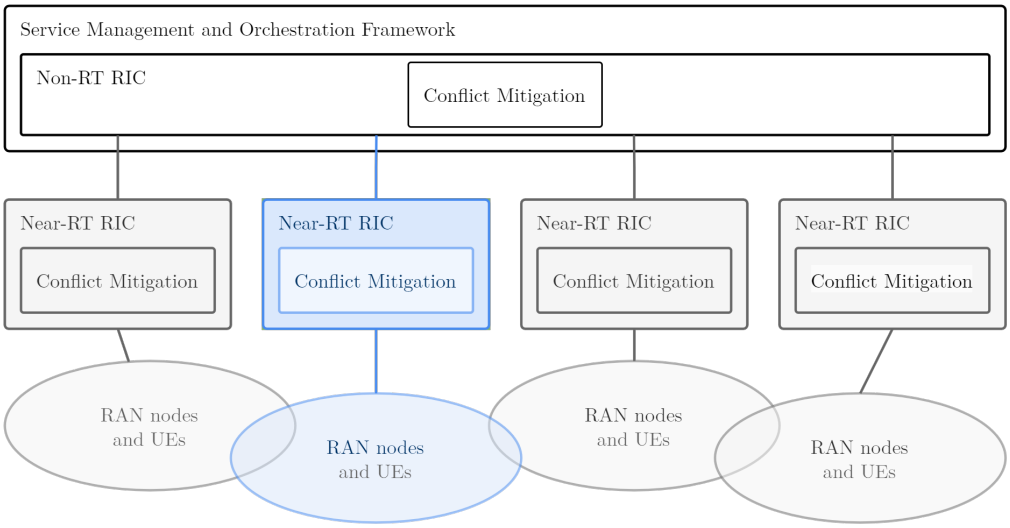

FIGURE 3.9: Challenge for conflict mitigation: limited observability of network control agents

Source: own elaboration

Guidelines for addressing the challenge

To solve this challenge, it may be necessary to develop advanced monitoring and analytics tools that can provide deeper visibility into the behavior of O-RAN nodes and RICs, even in situations where direct observation is limited. This could involve the use of ML algorithms that can automatically detect and diagnose potential issues based on network data, or the development of specialized monitoring tools that can capture detailed information about network behavior.

In addition to developing new monitoring and analytics tools, it may also be necessary to establish new protocols and standards for sharing network data and information between different O-RAN components to ensure that all network elements have the information they need to operate effectively. Alternatively, some required information sharing may be accomplished by defining new use cases for the existing O-RAN interfaces.

3.2.6 Summary of challenges and proposed solutions

Table 3.2 summarizes the challenges for conflict mitigation in O-RAN identified in above sections. For each challenge, general guidelines for addressing the issue are provided, alongside the specific solution proposed by the author. The final column references the sections where the proposed solutions are described.

3.3 Principles for conflict mitigation

To address all of the described challenges in O-RAN ConMit, a systematic and comprehensive approach is required to be implemented widely across the telecommunications industry. This approach should take into account the various aspects of the O-RAN standard and should be based on the following principles:

1. Optimization – ConMit capabilities of O-RAN must not worsen and, in majority of cases, should improve network performance compared to the situation without ConMit measures in place. While it may not be possible for any ConMit solution to not worsen performance or reliability from the perspective of all metrics, the trade-off between improved and deteriorated metrics must be as beneficial to the network as possible.

Table 3.2: Summary of challenges for conflict mitigation and proposed solutions

Challenge	Guidelines	Proposed Solution	Reference
Reliability of conflict detection mechanisms	Development of advanced algorithms (including ones based on AI/ML) for runtime detection of conflicts, standardized interfaces for exchanging network state between RICs	Conflict Detection & Resolution control loop	Section 5.4.1
		Guidance on AI/ML-powered CD Agent	Section 6.1
Maintenance of CM configuration in evolving network	Automated methods for updating configuration, adapting OSS/SMO to provide insight on control agents and conflict areas to network operators	Supervision & Adaptation control loop	Section 5.5
		Adaptive CR method via ACCoRD within CMF	Section 6.2
Optimal conflict resolution logic	Robust CR methods (including ones based on AI/ML) to derive optimal policies for resolution of conflicts	Simple rule-based prioritization and prioritization with cooldown methods	Section 5.4.5
		Robust adaptive ANN-based CR method	Section 6.2
Methodologies for testing and evaluation	Systematic frameworks covering multiple scenarios (topologies, traffic), standardized metrics	Novel evaluation method for CR methods	Section 5.4.7
		HW evaluation of CMF	Section 7.5
Limited observability of network control agents	Advanced monitoring/analytics tools (ML-based diagnosis), new protocols or use cases for existing interfaces to share ConMit data	Performance Monitoring component	Section 5.2.3
		Exchange of ConMit information between RICs	Section 5.2.5

2. Standardization – any introduced solutions, e.g., O-RAN architecture extensions and new protocols, need to be standardized to fit in the existing O-RAN landscape. It needs to be ensured that the issue of control conflicts is approached uniformly across the industry and the ConMit solutions and other software O-RAN components provided by various vendors are compatible with each other.

3. Automation – the operation of ConMit in O-RAN and its maintenance need to be as self-operating as possible. This includes evaluating and monitoring ConMit performance and reconfiguring ConMit to improve efficiency or adapt it to handle new control agents. Efficient automation in the context of O-RAN ConMit is likely to be enabled with ML and AI techniques, utilization of which are already envisioned in O-RAN's RICs.

4. Objectivity – the industry needs to develop objective metrics for evaluating the reliability of conflict detection and resolution methods and incorporate these metrics into the testing and evaluation processes that cover a wide range of O-RAN use cases and deployment scenarios. Furthermore, any ConMit solution implemented within O-RAN must be agnostic to vendor-specific biases in input data provided by control agents, or, ideally, should operate without requiring any such data.

5. Collaboration – the cooperation between MNOs, software vendors, and research institutions needs to be encouraged to share ideas and expertise in implementing efficient ConMit solutions.

3.4 State of the art

3.4.1 O-RAN Alliance specifications

The baseline for the discussion on the current state of the art in O-RAN ConMit are the specifications published by the O-RAN Alliance. As mentioned in the first section of this chapter, O-RAN specifications envision ConMit as a native function within the Non-RT and Near-RT RICs as the platforms hosting rApps and xApps [44, 45].

Conflict mitigation in the Non-RT RIC

As for the Non-RT RIC, the specifications until recently provided only minimal coverage of ConMit functionality. Effectively, the ConMit responsibility of the Non-RT RIC was limited to resolving conflicts in updates of A1 policies and EI jobs [44, 110]. To do so, the Non-RT RIC must monitor the policies and their enforcement status (i.e., whether they are successfully enforced by the Near-RT RIC or not), along with any EI jobs configured in the Near-RT RICs. In case of any conflicting update (including creation) of A1 policy or EI job in a Near-RT RIC is detected, it is the Near-RT RIC that communicates the conflict via *409 Conflict* HTTP error code [110]. The Non-RT RIC must then analyze the rejected conflicting request and consider sending a modified non-conflicting request.

Recently, the coverage of ConMit for A1 policies in O-RAN specifications was expanded. The O-RAN Alliance published a technical report on A1 policy ConMit in June 2025 [111]. The report first identifies three use cases for mitigation of A1 policy conflicts. The first one considers clash of concurrently enabled policies with conflicting goals. Two mutually exclusive groups of policy types are identified in this context:

- reserve policies, which reserve network resources to meet the optimization targets,
- release policies, which release network resources by offloading cells to meet the optimization targets.

In this use case, the A1 policy conflicts happen when policies from both groups are active at the same time in relation to the same scope (i.e., same UE, slice, cell, QoS).

The second use case described in the report addresses the situation where concurrent policies define a conflicting goal on different scopes, while both policies can be applied to a the same component. As an example, consider two rApps setting different targets (15 Mbps and 10 Mbps) for throughput for different scope (UE and slice). Since both of these policies can be applied to a UE connecting to that specific slice, it is not clearly defined which QoS target needs to be applied to that UE.

The third use case from the report covers lack of clarity in resources preference with conflicting policies. This case is mostly applicable in Traffic Steering (TS) policies, which operate through "SHALL" statements specifying target cells for UE handover. The conflict occurs when TS A1 policies define different target cells for specific UE and specific QoS, where both policies can be applied to a single UE.

Based on the described use cases, an example conflict detection flowchart for A1 policies is presented. It consists of two detection phases: first, overlapping scopes or resources in the policies

are verified, and then conflicts with policy objectives or resources are checked. Thanks to this logic, the ConMit is able to define sets of conflict-free and conflicting policies. The report presents detailed illustrated examples of the CD logic.

The report specifies A1 policy conflict avoidance in Non-RT RIC as functionality to prevent detected conflicts before the policies are applied in Near-RT RIC. Authors of the report envision that rApps will detect potential A1 policy conflicts either autonomously or via an R1 service offered by Non-RT RIC. Then, an entity within the Non-RT RIC can use A1 policy conflict arbitration service to mitigate the detected conflict; this mitigation can be triggered either by the rApp that detected the conflict or by a producer of A1 policy conflict arbitration service itself. The arbitration is then performed according to rules configured by the operator.

The simplest arbitration method for A1 policy conflicts presented in the report is to enforce the policies in order when the policy was created or updated. More robust mechanisms are also introduced to enable arbitration with the possibility to prefer certain types of policies over others. Specifically, the concepts of priority, preemption and preempt protection flags are introduced for A1 policies. The priority determines the preference of execution among a set of conflicting A1 policies. The preemption flag assigned to an A1 policy means that this policy can preempt the execution of another A1 policy. The preempt protection flag applied to an A1 policy counteracts the preemption flag by preventing this policy from being preempt by a policy with preemption flag. After arbitration, Non-RT RIC may notify rApps to retry provision of their policies once the situation changes and the conflicts are no longer expected. The report also provides several detailed examples showcasing how these mechanisms can be applied to express the policy preferences during conflict arbitration in Non-RT RIC. The report concludes with a set of recommendations for services related to A1 policy conflicts.

All in all, the technical report on A1 policy ConMit provides a quite detailed overview of mechanisms for detection, avoidance and resolution of conflicts in the Non-RT RIC. It assigns the responsibility for hosting and utilizing these mechanisms to the ConMit component within the RIC. However, it does not provide specific logic for how these mechanisms (like preemption and preempt prevention flags) should be used to optimally mitigate A1 policy conflicts. These details are out of the scope of O-RAN specifications and are up to the specific implementations to decide.

Other than this detailed technical report from O-RAN Alliance, the ConMit functionality of Non-RT RIC is not widely researched in the literature. The majority of research focus in O-RAN ConMit is centered at conflicts between xApps in the Near-RT RIC.

Conflict mitigation in the Near-RT RIC

As the interdependencies between xApps can be significantly more complex than A1 policy collisions, the ConMit mechanisms within the Near-RT RICs must be adequately more robust. O-RAN specifications provide insight into the ConMit functionality within the Near-RT RIC in document describing its architecture [45] and in a dedicated technical report [17]. O-RAN considers the problem of ConMit in the Near-RT RIC [17, 45] in three categories:

- Conflict Detection - detection of potential and actual conflicts (direct, indirect, implicit),
- Conflict Resolution - resolution of actual conflicts, including actions more complex than turning xApps off and on,
- Conflict Avoidance - provision of guidance and additional information to xApps to avoid future conflicts.

O-RAN specification for Near-RT RIC architecture [45] outlines two distinct mechanisms for ConMit operation:

- E2 Conflict Mitigation Guidance - xApps request the Near-RT RIC for guidance information regarding network control decisions. The guidance responses are then computed by the RIC based on collected network data and propagated to all relevant xApps (not only the requesting one). Guidance provided by the RIC can then be modified if network conditions change (e.g., due to new potentially conflicting control decisions).
- E2 Conflict Mitigation Assistance - Near-RT RIC detects conflict and requests xApps to provide assistance for conflict detection, avoidance or resolution process; the xApps responding to assistance may include dedicated helper xApp (such applications are called hApps [45]) or one of the conflicting xApps. The scope of assistance from helper xApps includes data collection and analysis. As for the conflicting xApps, the assistance data from them include information for computation of an avoidance solution and E2SM-related information relevant to the conflict. Based on data collected by the Near-RT RIC through assistance requests, it can mitigate the control conflict by performing any of the conflict detection, avoidance or resolution processes.

Implementation of these modes comes with significant challenges. Both require additional messaging between xApps and Near-RT RIC and, at the time of writing this thesis, the relevant messages are not specified by O-RAN Alliance [50, 57]. Furthermore, using these modes of ConMit operation requires significant effort to implement the logic for handling ConMit guidance and assistance data in xApps and the RIC. O-RAN specifications envision many various use cases for xApps, so the logic for processing ConMit guidance and assistance data is likely to be complex. The relevant ConMit communication and computation required in the two modes envisioned in O-RAN specifications will require significant time to be executed. Hence, implementing these modes is likely to be problematic due to the timeliness requirement of the conflict resolution decisions – the ConMit decisions must be taken as soon as possible to minimize the negative impacts of the conflict.

A more detailed overview of mitigation of control conflicts capabilities in Near-RT is described in a technical report published in October 2024 by O-RAN Alliance [17]. It addresses the topic by focusing on three areas: conflict detection, resolution, and avoidance. The report considers only the conflicts apparent in xApp actions using the E2 interface and does not focus on why these actions are taken (e.g., due to conflicting A1 policies).

In general, the topic of conflict detection is addressed in relatively the most detail in the technical report. Presented key issues for detection of conflicts in E2 operations relate to detection of potential conflicts, direct conflicts, and indirect/implicit conflicts. The potential conflicts were not previously specified in O-RAN specification [45]; the authors of the report consider probable conflicts which may happen in the future and can be detected before any E2 action is taken by xApps. Such conflicts may be determined by the Near-RT RIC's ConMit component by analyzing E2AP-level data about deployed xApps. This information includes E2 action type (POLICY/CONTROL) and identifies the target service by RAN Function and E2 Node IDs. However, this E2AP-level approach is considered sub-optimal as it may lead to many false positives, so adding analysis of E2SM-level information is proposed as the extension to achieve detection of direct conflicts. By comparing the specific E2SM-level control actions of many xApps, the RIC can confidently detect conflicts between decisions modifying the same parameters. Addition of such E2SM-level analysis increases complexity of the ConMit solution, as it may require understanding and managing of

multiple E2SM versions supported by deployed xApps. The report envisions that this E2SM-based CD may also be performed by a dedicated xApp using E2 Conflict Mitigation Guidance, instead of being an integral function within the Near-RT RIC's ConMit component.

The report also proposes a method for post-action detection of indirect and implicit conflicts. According to the report, a prerequisite for this detection is detection of a potential conflict between the conflicting xApps [17]. Additionally, a direct conflict must not be detected for the xApps for ConMit to post-action declare an indirect or implicit conflict. The ConMit component performing this detection requests additional conflict-related information from the xApps suspected of conflict and may subscribe to data related to operation of the xApps. Based on analysis of the collected information, ConMit may declare a control conflict.

A separate group of proposed CD solutions is based on preconfigured knowledge about dependencies between control parameters. These solutions are aimed to pre-action detect indirect and implicit conflicts. This operator-provided knowledge is expected to take form of a table specifying which parameters influence the same KPIs. These solution assume that xApps consult their control decisions with ConMit component via E2 Conflict Mitigation Guidance before executing them. Indirect or implicit conflicts are declared by the ConMit component once it identifies multiple xApps aiming to modify parameters related to the same KPI.

The report's findings about conflict resolution are vastly less robust in comparison to conflict detection. It provides only a single E2SM-level solution for direct parameter conflict resolution, introducing a mechanism for ConMit to negotiate a target value of the parameter by receiving information from conflicting xApps. This method can work for conflicts detected both pre-action and post-action. Depending on ConMit's method to compute the target value, xApps may be required to send only their preferred value of the parameter, or to include additional information, such as their utility functions required for game theoretic approaches like Nash Social Welfare Function (NSWF).

Similarly to conflict resolution, the report provides limited coverage of conflict avoidance. It envisions avoidance of conflicts only via E2 Conflict Mitigation Guidance coordination initiated by xApps expecting a potential conflict. The ConMit component within the Near-RT RIC can then assess the current situation and inform the xApp whether its action may cause a conflict and, if so, suggest how to modify the action to avoid it. Then, it is up to the xApp to either conform to the suggestion or not. This procedure is highly dependent on deployed xApps to properly implement it and follow ConMit's guidance.

Summarizing, O-RAN Alliance's technical report on ConMit in the Near-RT RIC outlines the issues for detection, resolution, and avoidance of conflicts between xApps [17]. It also provides some solutions to address these issues, but these solutions introduce many challenges themselves. Most importantly, as they are based on E2 Conflict Mitigation Guidance, the proposed solutions require xApps to be designed with ConMit coordination in mind and rely on the information they provide via exchange of the guidance information. This makes the solutions effective only when xApps properly implement the related procedures and provides accurate data to the ConMit component. Moreover, consulting each control decision with the ConMit component via E2 Conflict Mitigation Guidance procedure introduces significant coordination overhead before a conflict can be mitigated, leading to increased inertia in the Near-RT optimization of RAN performed by xApps.

3.4.2 Research on SON self-coordination

Research on SON's self-coordination function for mitigating conflicts between SON functions can act as a foundation for ConMit in O-RAN. However, because SON and O-RAN differ in

architecture, use cases, and RAN intelligence principles, findings from SON research rarely transfer directly to O-RAN.

In an early analysis of SON challenges, **Prehofer and Bettstetter [112]** investigate key principles and design paradigms in SON from the perspective of increased complexity of a highly automated heterogeneous network. This work was published in 2005, 4 years prior to inclusion of SON in 3GPP specifications. The authors highlight the tradeoff between introducing intelligent network automation and giving up control over its configuration. To make design of SON easier, four paradigms are proposed. First, the authors describe the idea of local coordination for achieving global targets. With no established centralized coordinator, the SON functions must individually take responsibilities for achieving target configuration goals. To deal with this challenge, the authors propose a divide-and-conquer approach with clustering the SON functions to clearly define the local views. Second, the article defines the paradigm of implicit coordination. The authors argue that perfect coordination often cannot be achieved and, hence, the best approach may be to contain the conflicts via tolerance of simple control conflicts, which are local and time-restricted. Implicit coordination is proposed for this purpose. Such coordination is based on nodes' observation of the environment instead of explicit exchange of information between the nodes. This approach is considered appropriate for localized coordination. Then, Prehofer and Bettstetter present the paradigm of minimizing long-lived state information in the network. It is the consequence of the two previously described paradigms – the less global coordination is needed, the less state information need to be kept up-to-date by the network nodes. The authors provide an example of using discovery mechanisms for learning information about other network nodes instead of relying on static global configuration. By following this paradigm, the network becomes more adaptive and immune to dynamically changing environment. The last design paradigm presented in the article considers usage of adaptive protocols. This is crucial for the SON functions to be able to react to changes in the network, including factors such as resource availability, user demands, and network node failures. As there is no global entity to signal changes in the environment, each network node needs to monitor the state of the network to gather this information and adapt its behavior accordingly. The article concludes with an example of how to design a SON function following all of the proposed paradigms.

Döttling and Viering [113] provide a broader and more up-to-date look on the state of SON and the challenges related to it. The article lists five main challenges for deploying SON. The first challenge considers the parameter dependencies between various SON use cases, providing handover optimization and load balancing as example. As various use cases may share relevant RAN configuration parameters, the SON algorithm must consider potentially conflicting optimization goals and compute them into a single target function based on current policy applied by the MNO. Next, the authors highlight the issue of information quality in decentralized SON deployments. They point out the tradeoff between improving information quality and increasing signaling overhead that must be considered in design of SON solutions. With limited knowledge about the neighbor cells makes it impossible for a SON function to detect all configuration conflicts. Furthermore, as a SON may operate using different vendors' equipment, standardization of information exchange protocols is needed to ensure consistency and reliability of the information. As a consequence of the two described challenges, the third one is the complexity of SON algorithm design. Due to the large scope of data describing the state of the network, the algorithm cannot realistically utilize a full search approach. Any trial-and-error approach is also not sensible for consideration in a real network.

As such, applying any researched solution in a real network may require significant adaptation. The fourth described challenge considers the stability and convergence of any implemented SON algorithm in presence of various coexisting mechanisms. This poses the requirement for any network node to obtain relevant configuration in a timely manner, e.g., after being provisioned in the network. Last but not least, the authors describe the challenge of sensibly evaluating the gains of specific SON solutions. They highlight the problem with defining a single evaluation metric that would quantitatively capture a solution's performance according to operator's requirements. The challenge of implementing a representative and efficient SON simulator due to scalability and complexity issues is also presented.

Research on SON includes efforts to provide a classification of conflicts possible various supported use cases. Defining conflict classes is a divide-and-conquer approach to ConMit. Having these types of conflicts clearly separated allows the researchers to consider the problem piece by piece instead of tackling the entirety of it at once. An example of such effort is the work by **Lateef et al. [114]**, where five conflict categories are recognized and applied to pairs of various SON functions:

- KPI conflict - various SON functions affect the same KPIs of a network node by controlling different configuration parameters,
- parameter conflict - various SON functions manipulate the same parameters of a network node; further categorized into output parameter conflicts (same parameter is modified) and input parameter conflicts (actions of one SON function influence parameters taken as input by another SON function),
- network topology mutation - addition or removal of a network node causes the optimal configuration to shift, requiring configuration readjustment for SON functions,
- logical dependency conflict - various SON functions operate based on objectives with logical dependencies,
- measurement conflict - SON function makes a decision based on a outdated measurement.

The authors classify combinations of the SON functions into these categories. For example, they label Coverage and Capacity Optimization (CCO) and Inter-Cell Interference Coordination (ICIC) function combination as output parameter conflict, and Mobility Load Balancing (MLB) and CCO as logical dependency conflict. As an example of the network topology mutation, MRO is provided as one of the SON functions affected by addition or removal of a network node. Multiple self-coordination mechanisms are identified as measures to mitigate the described conflicts in SON. These include algorithm coordination, autognostics function, decision tree logic, and many others. Lateef et al. consider the case of output parameter conflict between MRO and MLB, where both SON functions operate by modifying cell handover offset to meet different objectives. A self-coordination trigger-condition-action approach based on this scenario is put forward, with detailed logic for MRO and MLB defined. A set of 11 various actions are defined for both SON functions, each applicable in one of the cases considered in the algorithm, but no proof of efficacy of the proposed approach is shown. Finally, the authors conclude by providing a mapping between the conflict categories and the proposed coordination mechanisms to mitigate these conflicts.

Further work by the research team lead by **Lateef et al. [115]** expands on the topic by providing a soft classification approach, where 23 SON function combinations are assigned all applicable conflict types. The article then describes nested conflicts; these are conflicts which occur due to some preceding conflict. An example of a nested conflict is when conflict between MLB and CCO leads to the conflict between MRO and MLB. The authors highlight that these nested conflicts may require priority treatment from the perspective of SON conflict resolution, as low impact conflicts may lead to high impact ones. Finally, the hybrid self-coordination framework is presented, where multiple mechanisms are used in tandem for coordination between MRO and Energy Efficiency (EE) SON functions. In the considered case, the network nodes hosting MRO and EE functions exchange SON-related information through the X2 interface to coordinate when decisions are made by the functions. Then, one function may delay its control decision to ensure that effects of a prior decision are visible in the measurements taken as input parameters. This approach mitigates the measurement conflict. Furthermore, root cause analysis is perform at both network nodes to mitigate the logical dependency conflicts. Results of the root cause analysis is also transferred to a centralized server to enable further configuration corrections. The proposed solution is evaluated by comparison to an uncoordinated deployment with MRO and EE. The results show that the proposed solution significantly improves on the handover failure ratio by reducing it over two times, while simultaneously raising the average power consumption by approximately 10%. The results are replicable regardless of the total number of users in the considered scenario.

Other than the works that focus mostly on classification of control conflicts in SON, there is a large body of papers presenting various frameworks relevant to the issue of self-coordination. **Gelabert et al. [116]** investigate the interactions between SON control loops in distributed network nodes to propose a framework to evaluate and coordinate these cross-function dependencies. In this case, the proposed "framework" means a structured approach to coordinating SON functions (and not introduction of a logical component and relevant procedures tasked with resolving conflicts). The authors consider a case of conflicting Admission Control (AC) and RAT Selection (RS) functions. The former is tasked with radio resource allocation to new users in a cell, while the latter decides which RAT is assigned to serve any user. Both of these functions can be considered as responsible for load balancing, although with different general goals. As such, these is a clear control conflict between them. The approach put forward by Gelabert et al. is to model the system as a discrete-time Markov chain and consider probability of transitions between states based on stochastic modeling of network behavior (i.e., events such as users connecting to the network) and logic of considered SON functions. Each state is defined by number of users connected to the two collocated RATs deployed as part of the considered network. The paper provides an analysis of impacts of various parameters of AC and RS on grade of service. Finally, using the optimal parameters derived in the previous step, five combinations of SON mechanism configurations are compared in terms of average unbalance factor and grade of service. Results show that combination of the two SON mechanisms working concurrently slightly outperform the cases where only one of them or none is active.

Another example of research on frameworks for SON self-coordination is the work by **Schmelz et al. [117]**. The authors proposed their own classification of conflicts, which is simpler than the ones proposed in other works dedicated to this topic [114, 115]. Specifically, the article describes

two types of control conflicts: control parameter conflict (related to conflict in directionality or magnitude, which could be mapped to output parameter conflict from [114], and observability dependency conflict, which could be mapped to input parameter conflict from [114]. The logical entity responsible for self-coordination in the proposed framework is the SON Coordinator, which manages SON functions and their control decisions based on MNO-defined high-level performance objectives. The network operator can also provide policies specific to the conflict resolution process, such as limitations to allowed parameter ranges or allowed periodicity of parameter changes. The coordinator entity also allows the operator to set up reports about network performance and custom alarms for undesired events. This intent-driven approach requires that SON Coordinator implements robust logic to be able to manage the SON functions' decisions within the constraints resulting from the MNO's policies. The proposed SON Coordinator hosts four functions:

- Policy function - performs avoidance of conflicts by computing MNO's global targets into SON-function-level policies and setting constraints for parameter ranges,
- Alignment function - performs detection and resolution of conflicts, manages network configuration in critical network conditions detected by Guard function; this function is split into subfunctions Activation (for cause analysis and SON function management) and Arbitration (for detecting and resolving conflicts, including temporarily locking parameters to given value),
- Autognostics function - collects performance, fault and configuration management info for the SON system; this function can also filter out unreliable data to ensure consistent data for all recipients,
- Guard function - triggers fail-safe corrective actions when critical network conditions are detected based on data provided by Autognostics function.

Specific self-coordination activities are managed with the assistance of SON Coordination database, which is controlled by the Alignment function. The work also describes various SON function arrangements, where SON Coordinator can assign coupled functions into groups, which are managed by local instances of Policy and Alignment functions. The proposed framework is evaluated based on a case with conflicting MRO and MLB SON functions. A number of metrics describing network performance are compared between five cases: without SON functions, with MRO only, with MLB only, with both MRO and MLB, and with both functions with coordination enabled. The implemented Alignment function for resolution of conflict between MRO and MLB by suspending modification of handover hysteresis by MRO for any cell that was often overloaded. The results show that introduction of the coordination between SON functions allows to significantly reduce the call drop ratio with slight increases to handover ping-pong ratio and number of unsatisfied users.

Iacoboaiea et al. [118] focus on detection and diagnosis of control conflicts in SON and provide a framework dedicated to this task. The authors limit their considerations to two simple types of conflicts: parameter conflicts (where SON functions change the same parameter; such conflict could be mapped to output parameter conflict from [114]) and measurements conflicts (where parameter modified by one SON function affects measurements used as input by another function; such conflict could be mapped to input parameter conflict from [114]). The proposed conflict diagnosis framework considers SON functions as black boxes, meaning that the framework does not have any insight into what is the internal logic of the SON functions. The diagnosis

algorithm employs a naïve Bayes classifier, which computes conditional probabilities under the assumptions that observed features are independent given each cause and that only one cause applies at any time; then, the most probably cause is selected as the resulting diagnosis. For evaluation of the proposed framework, the authors consider a SON with central macro cell and multiple pico cells within. Three consecutively running SON functions are taken into account:

- MLB - balances pico cell load by manipulating CIO,
- MRO - reduces average percentage of too late and ping-pong handovers between pico cells by modifying handover hysteresis,
- ICIC - maintains fairness among network users by manipulating number of Almost Blank Subframes (nABS) based on ratio between average throughput of users handled by macro cell and average throughput of all users handled by pico cells that would be serviced by the macro cell if hysteresis and CIO were set to 0.

The authors define matrices describing input and output parameters of the three SON functions, along with a matrix showing which parameters have significant impact on considered key network metrics, such as ratio of too late or ping-pong handovers; this last matrix is compiled based on expert knowledge of the authors. To determine which SON functions may conflict with each other, the matrices are multiplied using Boolean algebra. The result matrix shows that MLB impacts MRO and ICIC, and ICIC impacts MLB. The work presents evaluation of the diagnosis algorithm various precision of the detection based on 12 cases, where causes are defined as combination of conflicting SON function (MLB, ICIC), parameter settings (trigger thresholds, step sizes), and magnitude of parameter change (low, medium, high). As expected, the diagnosis is the most precise when limiting its scope to only detecting the conflicting function, and the least precise when detecting set of function, setting and change magnitude. The worst presented case is around 55% detection success for detection of ICIC conflict with step size setting and high magnitude of change.

In comparison to previously described self-coordination framework efforts, **Tall et al. [119]** approach the topic with focus on distributed coordination. The authors envision a generic coordination mechanism for multiple various distributed SON functions. Operation of SON mechanisms and their interactions, along with the proposed SON coordination mechanism, are modeled mathematically. The proposed coordination algorithm is based on control theory and Lyapunov stability. A case with 19 omnidirectional cells is considered, with the 3 following SON functions activated in 7 centrally positioned base stations:

- Load Balancing - optimizes transmission power of pilot signals to balance load between cells,
- Blocking Rate Minimization - adjusts admission threshold to reach a desired blocking rate value,
- Outage Probability Minimization - handles transmission power of data signals to reach a desired outage probability.

Network performance metrics in cases with and without coordination are analyzed. The results show that without SON coordination, as the network simulation progresses in time, the performance metrics deteriorate. Introduction of SON coordination enables stabilization of the metrics on desired levels. Specifically, base station load for three plotted cells is kept at 50-65% level instead

of 70-85% without coordination. In addition, coordination keeps outage probability at 20-25% for cells #2 and #3, while in case without coordination it soars to 90-100%. On the other hand, outage probability for cell #1 is worse in case with coordination (5%) in comparison to the case without coordination (near 0%), but the difference is not as drastic as for the other two cells. Simultaneously, blocking rate for cell #1 without coordination is significantly larger (2-3%) than blocking rate for all other plotted cases, where it is generally under 1%.

Banerjee et al. [109] propose an alternative approach to resolving conflicts between SON functions via introduction of the concept of Cognitive Autonomous Network (CAN), which replaces SON functions with Cognitive Functions (CFs). A CF does not work according to some predefined logic (like a conventional SON function) but infers its policy based on its learning experience. Hence, a rule-based method for coordination is not suitable. As a remedy to this issue, the authors propose Cognitive Bargaining Mechanism (CBM). This mechanism aims to enable conflict resolution for all types of conflicts, regardless of the number of conflicting CFs and the number of simultaneously occurring conflicts. The authors utilize the Nash Social Welfare Function (NSWF) as the algorithm to compute a compromise between conflicting CFs. Due to the properties of the NSWF, objectives of all CFs have the same weight and are treated as equally important. The considered system model assumes that all CFs provide the controller executing CBM algorithm with optimal ranges of parameter values. Each update of optimal parameter range provided by any of the CFs triggers the controller to request all CFs who use the same parameters to provide their updated optimal parameter ranges. When all required information is collected, the controller recalculates the configuration. This approach depends on implementation of CFs to be able to provide optimal parameter ranges (and generally cooperate with the controller) in a timely manner; otherwise, the computed configurations may not suit dynamically changing network conditions. The authors consider three CAN models, each defined as to verify various combinations of all three considered categories of conflicts:

- configuration conflicts - on input or output parameters,
- measurement conflicts - where actions of one CF influence measurement considered by another CF,
- characteristic conflicts - direct characteristic conflict or logical dependency conflict.

In totality, there are 13 CFs deployed across the considered scenarios. Numerical analysis shows that the proposed algorithm is able to find optimal configurations regardless of the conflict scenario. It is important to point out that the analysis is based on artificial CFs operating on generic metrics and may not exactly map to a more realistic network model.

A framework with broader functionality scope is proposed by **Bag et al. [120]**. Their Generic engine for Self-Healing, self-Optimization and self-Coordination (G-SHOCC) uses FL to train a Deep Neural Network (DNN) and achieve coordination between the three SON functions considered in the scenario: CCO, ICIC, and Cell Outage Compensation (COC). The authors propose an FL-based approach as a way to reduce scope of data exchanged between distributed entity and a centralized controller and to mitigate security concerns that MNOs may raise when asked to provide large datasets to a 3rd party vendor's controller software. A key element of the proposed

framework is the Local Multi-Objective (LMO) KPI, which provides a common metric by combining goals of the coordinated SON functions. This KPI is what the implemented DNN predicts for the possible sets of parameters. Based on the prediction results, the parameter set expected to provide the best network performance is applied. The authors evaluate the implemented solution using computer simulation of the SON with a centrally trained DNN model and a model trained using a decentralized FL approach. An additional case of statically configured best observed parameter set is included in the results to evaluate validity of the configuration in normal and faulty network conditions. The considered simulation scenario includes a fault in a central RAN node to include verification of self-healing capabilities of COC; this fault appears in the middle of each simulation run. The authors indicate a 45% improvement in LMO KPI when comparing online running models (both centrally trained and trained via FL) and the static configuration. In addition, the online models decrease Reference Signal Received Power (RSRP) outage rates by over 15% in comparison to static configuration. In these two considered cases, the performance of centrally trained and FL-trained models are nearly identical. However, considering mean network Signal to Inference plus Noise Ratio (SINR), the model trained using FL improves on static configuration by over 14%, while the centrally trained model only by nearly 9%. It is important to highlight that the SINR plots for online models fluctuate significantly, while the plot for static configuration changes smoothly over time.

Bandh et al. [121] present a short paper on an experimental SON function coordination scheme driven by policies. The authors consider a SON with two CCO functions: one operating on transmission power, and the other on remote electrical tilt of antennas. The method tested in this short paper is a method developed as part of other works by some of the authors [122, 123]. The coordination mechanism works by defining decision trees for relevant SON function types, where data from other coexisting SON functions are considered during configuration. This approach requires definition of exact logic how the coordination works. The paper shows how coordination between the two CCO functions fill in two coverage gaps in the simulated area, which were present in the case without coordination.

Tsagkaris et al. [124, 125] in their papers consider a SON coordination solution implemented as part of a Unified Management Framework (UMF). This framework operates on Network Empowerment Mechanisms (NEMs), which in case of SON represent the SON functions. One of three core functions of UMF is Coordination, which is the main focus as part of these articles. The authors consider a case of a SON with two functions: MLB and Backhaul Resource Allocation (BRA). The latter works by allocating portion of operation time of a backhaul link to relay traffic from other RAN nodes instead of the usual task of handling backhaul for user traffic. UMF coordinates the SON functions by activating them with the same time periodicity and static learning rates. The applied coordination balances load of links between backhaul and relay traffic at approximately 60% load for the worst links. Simultaneously, with enabled coordination the throughput of users located at cell edge increases from around 0.5 Mbps to over 2.0 Mbps.

Cinemre et al. [126] investigated utilization of various bargaining solutions in coordination of SON functions tasked with optimizing antenna tilt. The authors consider a network following

the Intent-Based Network (IBN) concept and employ the following bargaining solutions to infer optimal antenna tilt: Nash Bargaining Solution (NBS), Weighted NBS (WNBS), Kalai-Smorodinsky Bargaining Solution (KSBS), and Shannon Entropy Bargaining Solution (SEBS). Two SON functions deployed in the network, ICIC and CCO, directly conflict with each other by modifying the same parameter (antenna tilt) with different optimization goals. Evaluation of each considered bargaining algorithm is done based on a disaster recovery scenario, where one eNodeB must provide service to users initially served by two other, recently turned off eNodeBs. With the transmission power fixed, the antenna tilts are then optimized by ICIC and CCO within the range from 0 to 15 degrees with 1 degree step. Two metrics are considered for comparison of simulation runs using the various bargaining mechanisms: SINR and Channel Quality Indicator (CQI). Each solution computes a different optimal antenna tilt setting, with WNBS configured to prioritize CCO's KPI. Jain fairness index was used to select the best solution and KSBS emerged as the most equitable.

Another approach considering conflicts in antenna tilt adjustments was later proposed by **Akbas et al.** [**127**]. The solution is a rule-based mediation approach aiming to reconcile two optimization targets. The authors consider the control conflicts from the perspective of trade-offs between KPIs. As an example, they define a direct trade-off as a situation where improving on a KPI negatively impacts another one in a direct manner. Such a relation is observed between network capacity and coverage. These two KPIs are utilized by two DQN-based antenna tilt optimization agents implemented to evaluate the solution introduced in this article. The proposed conflict mediation algorithm ingests data about rewards considered by the deployed ML-based agents and prioritizes the tilt setting computed by the agent declaring a higher reward value. The authors declare that the implemented mediation scheme enables selection of the optimal antenna tilt setting, but do not present any specific values that could allow to evaluate the case with mediation against the cases without it.

Ali-Tolppa and Tsvetkov [**128**] analyzed conflicts between MRO and Remote Electrical Tilt (RET) SON functions to develop solutions for SON verification and coordination. The former is described as a method for automatic validation of network configuration based on observed degradations in network performance metrics. Upon detection of a significant degradation in one of the network areas, the verification method triggers configuration rollback actions for that area. Network performance is observed post rollback to ensure that the recovery actions worked as intended; in case the performance metrics do not improve, verification method analyzes further potential causes of the deterioration. The proposed coordination mechanism is called Optimistic Concurrency Control (OCC), which works under the assumption that multiple agents can perform transactions on a data set without the requirement to acquire an exclusive lock on the data. This is done by validating that no other transaction has modified the contested data set; if a clash is detected, then the changes are rolled back. The coordination process is timed according to cadence of performance metric data provisioning to the centralized SON coordinator. The coordinator categorizes conflicting functions into these in hard conflict (which must not be allowed to run in parallel when conflicted) and these in soft conflict (which can be run in parallel only if SON verification is in place). In addition, the coordinator algorithm dynamically adjusts the number of SON functions running in parallel each processing period. The implemented method was evaluated by the means of simulation scenario. In the considered case, one of the cells in a simulation area

with 12 eNodeBs is configured suboptimally, i.e., its electrical tilt is set to value of 15 degrees, with 5 degrees being the optimal setting. The three considered cases (no coordination, policy-based coordination, coordination with OCC) were evaluated based on CQI and handover success rate KPIs. The results show that CCO generally achieves best results out of the three evaluated approaches, with improvements in CQI and handover success rate for the degraded cell. As for the neighbors of the degraded cell, OCC achieves the best handover success rate, but lack of coordination allows for slightly higher average CQI throughout the majority of the simulation run.

One of the most commonly considered cases of network control conflict in SON is the indirect conflict between MRO and MLB SON functions. **Liu et al. [129]** envisioned a scheme to resolve conflicts between these two SON functions, which limits the range of changes allowed to one of the control parameters. The authors consider the impacts of MRO and MLB on handover boundaries of neighboring cells and how specific handover boundaries may lead to Radio Link Failure (RLF) due to too late or too early handover. Mathematical analysis of the relations between handover setting lead the authors to the conclusion that a promising approach would be to limit changes done to CIO by MLB. The implemented scheme considers a set of limitations to MLB, which is based on current handover conditions in the neighboring cells; one of the cell neighbors triggers the optimization process and calculates the target settings. Then, the leading base station requests the other one to adjust the CIO value and, upon acceptation of the request, updates its own CIO. The proposed algorithm is evaluated by comparison with a conventional method without restrictions to MLB. The metrics considered as part of the comparison are ratio of RLF, ratio of ping-pong handovers, number of blocked and dropped connections, and number of operations done by MLB. The proposed scheme generally ensures significantly better results than the non-restricted method regardless of movement speed of UEs, with the differences increasing as number of UEs in simulation topology increases.

The approach for resolving conflict between MRO and MLB presented by Liu et al. [129] (denoted as original solution in this paragraph) has been improved in later research by **Huang and Chen [130]** to find optimal values of CIO instead of restricting its allowed range. The authors propose to additionally consider load distribution of the relevant cells; this requires additional information exchange between the conflicting eNodeBs. The proposed scheme slightly improves on the original method in terms of RLF and ping-pong handover ratios. More significant improvements are observed for average cell throughput, ratio of overloaded cells, ratio of blocked calls, and Jain fairness index. As in the works by Liu et al. [129], differences between methods become more visible as number of UEs in simulation topology increases.

Another approach to the MRO and MLB conflict is provided by **Li et al. [131]**. It is again compared to the solution described in [129] (denoted as original solution in this paragraph). The method proposed by the authors of [131] is named optimal conflict avoidance; it considers movements of UEs to dynamically set time-to-trigger (TTT) for specific UEs and target range for eNodeBs' CIO setting. Comparison to the original solution is performed based on total cell throughput, ratio of blocked calls, and ratio of RLFs. In all considered metrics, the proposed solution provides better results in comparison to the original solution. In these results, the differences between performance

of methods do not change significantly change as number of UEs in simulation topology increases. It is worth highlighting that steering of handover parameters on UE level would drastically increase processing demands in comparison to the original method.

Zia et al. [132] came up with a policy-based method for conflict resolution between MRO and MLB SON functions called MLB-Static. The aim taken by the authors is to combine the goals of the two functions into one, and hence remove control instability caused by their conflict. The proposed algorithm runs in each eNodeB to find target CIO settings on a per-UE basis. During computation of the target settings, requests to hand over UEs from neighboring cells are taken into account to avoid conflict. Furthermore, the algorithm considers modification of CIO for users who appear as static. This is to avoid changing of MLB-related handover parameters for high-speed UEs. Effectively, only MRO optimizes handover parameters for these UEs. Evaluation of MLB-Static is done against one case with only MRO and another with MRO and MLB without coordination. When comparing plots of gain (represented as ratio of unsatisfied UEs) and cost (represented as a handover aggregate performance metric combining handover rate, ping-pong handover rate, RLF rate, and RLF rate), it is clear that the proposed MLB-Static algorithm improves on both handover performance and user satisfaction in two cases with different UE movement speeds.

Yet another look at the case of MRO (referred to as Handover Optimization (HOO) in the paper) and MLB conflict is provided by **Muñoz et al. [133]**. The authors consider the two functions conflicting over control of handover margin (HOM) parameter (which works similarly to CIO). The proposed approach works by limiting the target parameter range for the functions to avoid extreme value settings in these two specific scenarios: high-mobility case (when handover ratio exceeds 5%; MRO's setting is limited) and high-load case (when cell blockage ratio exceeds 2%; MLB's setting is limited). Three handover performance metrics are analyzed: call blockage ratio, call drop ratio, and handover ratio (i.e., number of handovers in relation to number of successfully finished calls). An additional composite utility score describing handover performance is defined, namely the U figure; it combines these three metrics into a single scalar value. Performance of the solution is evaluated in comparison to cases with no SON and with no coordination of MRO and MLB coexistence. Simulation results show that uncoordinated operation of both functions lead to significantly high value of call drop rate, while complete lack of SON leads to elevated call blockage rate. Influence of the proposed solution on the considered performance metrics depends on configuration of the limitation thresholds of the coordination solution, but generally is able to provide a better tradeoff between the metrics. For example, the coordination configurations significantly limiting handover parameters enable to decrease cell congestion and avoid large increase of handover ratio. The optimal configuration is chosen by evaluating performance in both high-load and high-mobility cases, utilizing the composite U figure to quantify the overall trade-off between the individual metrics. The chosen configuration limits MRO's setting significantly more than MLB's.

A DRL framework for ConMit between SON functions is proposed by **Tu et al. [134]**. The authors consider the conflict between MRO and MLB during optimization of the handover procedure. Based on analysis of the issue and some of the existing works utilizing ML in SON self-coordination, usage of DRL is envisioned as the solution. The proposed approach works by compiling the

coordinated operation of MRO and MLB into the following sequence of actions: detect load imbalance, perform load balancing using MLB, use DRL agent to offload select UEs to small cells, obtain feedback to train DRL agent, detect mobility resilience problem, perform handover optimization using MRO. It is envisioned that the RL-based coordination agent shall observe its performance and adapt to the changing network conditions. The authors only described the concept and did not present any results backing the appropriateness of their proposed approach.

Similar considerations regarding ML-enabled SON self-coordination are presented by **Stamatelatos et al. [135]**. Their analysis focuses on how ML techniques can enable intelligent ConMit in SON. Their proposal for using ML in self-coordination envisions deployment of ML-powered coordination entities for self-configuration, self-optimization, and self-healing SON functions. SON functions raise execution requests to the relevant coordinator entity, which then decides how to apply these requests to the network. The authors do not present any simulation results to evaluate the proposed scheme.

Iacoboaiea et al. [136] put forward a solution for SON coordination powered by ML, named SONCO. Specifically, it utilizes an RL Q-Learning algorithm to resolve conflicting settings for CIO provided by MRO and MLB. The authors argue that such approach can adapt its decisions based on effects of past decisions, which should provide improvement over conventional static-rule-based approaches. The proposed solution is deployed in a centralized SON coordinator entity. It is assumed that the deployed SON functions forward their decisions through the coordinator in a synchronized manner instead of executing the control decisions independently themselves. The RL algorithm evaluates the taken actions by summing up weighed happiness indicators of the SON functions. The authors consider three metrics to compare the various configurations (i.e., weights of SON functions' happiness in the reward equation) of the solution: average cell load, average number of too late handovers, and average number of ping-pong handovers. The results show that providing more priority to MRO helps reduce average number of too late handovers. Simultaneously, increasing MLB's weight improves on the load balancing capabilities of SON. Despite being mostly influenced by handover hysteresis controlled by MRO, average number of ping-pong handovers also generally decreases with weight of MRO. Unfortunately, the authors do not provide comparison to a baseline algorithm or a case without coordination or show how implementation of RL improves the solution's efficiency as it gains experience.

Coordination between RL-based SON MRO and MLB functions is investigated in an article by **Mwanje and Mitschele-Thiel [137]**. Their solution to the problem is utilization of Concurrent Cooperative Games (CCG). This approach introduces communication between SON functions to learn how given function impacts other functions in the same and neighbor cells. The authors limit the SON functions to work under the restriction that at the time there are no two functions that simultaneously affect the environment (maximum one SON function works in one cell in the network at any given moment). Furthermore, the solution requires that there is an aggregate reward indicator defined to capture quality of the SON functions' operation; this indicator is envisioned as an operator-defined metric aligned with the operator's policy intents. Evaluation of the solution is performed for the case with coordination for SON functions operating within a single cell and does

not consider impacts on the neighbor cells. The authors observe that uncoordinated concurrent operation of MRO and MLB leads to deterioration of handover performance even in comparison to static default settings. Simultaneously, introduction of CCG for coordination between functions allows to reach a compromise between optimization goals of the two functions. As one function considers decisions of the other as part of its decision-making process, the control decisions taken by the respective algorithms have less negative impact on the observed handover performance indicators. The authors highlight that the efficiency of the solution is highly dependent on the proper definition of the aggregate objective functions.

Moysen et al. [138] define a self-coordination framework that utilize ML techniques to compose multiple objective functions into a single optimization target. Similarly to some other works described above, each SON function directs its control decisions towards the centralized self-coordination framework instead of executing them on its own. The authors utilize a multi-objective optimization method called Non-dominated Sorting Genetic Algorithm II (NSGA-II). It works by mutating and combining solutions within a set during multiple evolution episodes. NSGA-II, in comparison to other genetic algorithms, applies non-dominated sorting and crowding distance to effectively identify solutions providing the best trade-off between considered objective functions. A prediction model trained using data about past experience of SON is fed into NSGA-II to find the best configuration of cell-level handover parameters. To evaluate the proposed solution, MRO and MLB are ran independently, and then with coordination using the proposed scheme. From the configuration sets provided using NSGA-II, one of the optimal sets is selected as the representative for comparison against non-coordinated SON functions. The solution is evaluated based on average cumulative distribution function of network-level average throughput. The authors conclude that the configuration derived from operation of the proposed scheme provides the best performance trade-off in comparison to MRO and MLB operating by their own.

A zero-touch ML-based approach for coordination of conflicts between MRO and MLB is proposed by **Preciado Rojas et al. [139]**. This centralized solution aims to learn relations between deployed SON functions and does not require knowledge about internals of the functions. The functions are modeled as RL agents using Q-Learning. Data gathered by SON functions is transferred to an ML environment, where it is processed into a model of dependencies in the considered network. The authors consider two variants of algorithm combinations for use in their solution: Uniform Manifold Approximation and Projection (UMAP) with Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) (denoted as U+H), and Self-Organizing Maps (SOM) with hierarchical agglomerative clustering (denoted as SOM+HC). For evaluation of the variants of the solution, other than the cases where SON functions work individually and together without coordination, two additional baseline algorithms are considered: Temporal Separation during Learning (TSL), where functions operate in turns during learning phase, and Concurrent learning with Spatial Separation (CSS), where functions in neighboring cells cannot work at the same time. The solutions are compared based on three metrics observed over simulation time: a handover aggregate performance metric (which weights ping-pong handover rate and rates of RLF due to too early and too late handover), rate of unsatisfied users, and a global KPI computed by the model. First, the results show that U+H and SOM+HC variants excel in terms of handover aggregate performance metric convergence to a stable, low level; especially the

SOM+HC solution reaches the target metric level almost immediately as the simulation starts. Over time, other solutions reach similar levels of handover aggregate performance metric, but none exceeds the efficiency that SOM+HC establishes early on. For the rate of unsatisfied users, the U+H solution variant (along with Q-Learning-based MLB operating alone) reaches the lowest level the earliest; in this case, SOM+HC converges similarly quickly, but to a slightly higher level. Both considered solution variants show similar, and best in comparison to the baseline methods, performance for the global KPI metric.

Another proposition for self-coordination in SON using ML techniques is presented in an article by **Lee et al. [140]**, where the conflict between MRO and MLB is modeled as a Markov decision problem and the DRL algorithm is employed in a central controller to solve it. Similarly to some previous solutions described above, the framework works under the assumption that SON functions communicate their intended control decisions to the centralized coordinator entity to apply them in the network upon receiving the coordinator's approval, instead of executing the decisions immediately with no coordination. The implemented DRL approach is Double DQN, where two Q-Learning based ANNs work concurrently - one for choosing the best action and the other to evaluate that action. The proposed scheme, called SLC2, is compared to non-coordinated solutions (only MRO, only MLB, MRO and MLB) and coordinated solutions (among others, MLB-Static [132] and SONCO [136]). Comparing convergence time in small- and large-scale networks, SLC2 shows the quickest convergence to target policy; furthermore, it remains stable after converging, which cannot be said about all tested solutions. Further evaluation based on average throughput per UE and total count of RLF and ping-pong handovers in small- and large-scale environments also show the best results for the proposed SLC2 solution.

Vlacheas et al. [141] look at the control conflict between ICIC and CCO SON functions and present a solution for self-coordination. The considerations start with definition of various conflicting interactions between SON functions. The authors define three interaction classes: parameter interaction (same as parameter conflict in [114]), metric interaction (same as KPI conflict in [114], and SON interaction (same as logical dependency conflict in [114]. Then, the interactions between ICIC and CCO are analyzed and classified according to the proposed categorization. The conflict coordination problem is formulated as a weighted multi-objective optimization problem, where weights are defined by the MNO. The target resource allocation matrix is computed by solving the optimization problem. The authors evaluate various weight configurations by plotting average user throughput, total received interference, average spectral efficiency, and number of unsatisfied users. It is observed that with weight 0.4 assigned to ICIC and weight 0.6 assigned to CCO, the network operates the best in terms of the considered metrics.

3.4.3 Research on O-RAN conflict mitigation

With the first relevant articles leading back to 2023, the research on ConMit in O-RAN is at a relatively early stage. As a result of this, there are multiple significant research gaps. On the other hand, the field shows rapid growth and clear trends are already visible in the published works. The research publications on the topic have kicked off with foundational developments of framework for ConMit in O-RAN and analysis of its main challenges [1] (as described by the author in Sections 3.1 and 3.2), following into many proposals of specific solutions for CD and CR. The

developed solutions often are ML-native and assume that the conflicting network control agents (i.e., applications) may use ML techniques. Simultaneously, some of the proposed ConMit solutions introduce AI/ML as part of logic for CD and CR computation. As the field of ConMit in O-RAN is on a relatively low development and discovery level, nearly all of the existing works describe unique approaches.

As O-RAN aims to enable robust RAN intelligence with the possibility to deploy many applications from many vendors, it is crucial to ensure that the deployed solutions work without negative impact on the network performance and reliability due to conflicting control decisions. One of the ways to manage the conflicts between deployed xApps is to introduce a systematic solution that is tasked with conflict detection and resolution. Due to this openness of O-RAN (various applications provided by various vendors), it is important that any ConMit solution works regardless of the types, vendors, and specific logic of the deployed xApps. Furthermore, to ensure compatibility, the ConMit solution shall not introduce any new requirements towards the implementation of the xApps (e.g., require xApps to share specific information outside of what is defined in O-RAN standard). While O-RAN technical specifications envision ConMit functions within Near-RT and Non-RT RICs and provide their high-level overview, specifications leave their implementation details and internal logic undefined.

Conflict Mitigation Framework

The first foundational work to define a comprehensive framework for detection and resolution of conflicts in O-RAN was published in 2023 by the author [2]. It introduced the concept of CMF deployed in the Near-RT RIC to enable mitigation of all types of control conflicts between xApps. CMF hosts dedicated logical components tasked with managing the ConMit process (including CD Agent and CR Agent) and is able to detect direct, indirect and implicit conflicts. Detailed descriptions of CMF and related procedures are provided in Section 5.4.2.

CMF can host any conflict resolution logic, including AI-powered algorithms. The article that defined CMF [2] proposed a simple rule-based method for conflict resolution that modifies conflicting control decisions. This method, called prioritization, works by prioritizing one of the conflicting xApps and rejecting control decisions made by any other xApp in case of conflict. It was evaluated in an O-RAN-certified CMF-enabled testbed [4] in works by Sultana et al., coauthored by the author. This work on hardware-based validation of CMF with prioritization method is further described in Chapter 7. The prioritization concept was expanded on by further works by the author [5], where extension of this conflict resolution method was considered for mitigating indirect conflicts. Specifically, the approach was improved by introducing a cooldown period. During this period, the non-prioritized xApps cannot make any decision regarding the control target, regardless whether this decision would result in a conflict or not. Both of the described methods are further detailed in Section 5.4.5.

QACM in CMF

Another rule-based method for conflict resolution in CMF named QoS-Aware Conflict Mitigation (QACM) was proposed by Wadud et al. [22]. The proposed method is significantly more complex in comparison to the prioritization solutions presented in [2, 5]. It builds up on the authors' previous works on game theoretic CR approaches utilizing NSWF and Eisenberg-Gale (EG) algorithms [18], which did not follow the CMF procedures, despite architectural similarities. QACM is designed to ensure that conflicting xApps reach their QoS targets despite the control decision modifications

introduced by ConMit. This is a valuable feature for a CR method, which in corner cases may completely override the intents of xApps' decisions.

The conflict resolution approach of QACM is to compute an optimal value of a control parameter for which conflict was detected. Effectively, the proposed CR algorithm ingests conflicting control decisions and processes them into a single decision, which should in principle satisfy the conflicting xApps. The optimization objective is defined as follows: minimize sum of scaled differences between the values of the relevant KPIs and their QoS thresholds subtracted by a squared sum of binary QoS satisfaction indicators. To make such approach possible, QACM must be able to compute distance between KPI and QoS threshold for multiple KPIs, which may have different units. This requires to define utility functions to bring the considered KPIs into a single range of scalar values; Wadud et al. chose to employ Z-score normalization over min-max normalization for this purpose to retain statistical distribution characteristics of the original KPI data.

To combat exponential growth of computation complexity in cases with large number of conflicting xApps, QACM envisions a heuristic algorithm to compute optimal values for control parameters. It solves the optimization problem by iterating over all possible parameter values from total range of values derived from xApp-declared optimal ranges. For each possible parameter value, the predicted utility function values requested from conflicting xApps are used to determine distance from target KPI thresholds and track whether satisfaction targets are met via binary satisfactory indicators. Total cost of given parameter value is calculated by summing the computed distances and subtracting squared sum of binary values satisfactory indicators; the computed total cost is then compared with the lowest total cost calculated so far. Once the algorithm iterates through all parameter values, the lowest cost parameter value is chosen as the optimal.

QACM requires that the CMF architecture shown in [2] is appended with information regarding cell-level allowed ranges of all control parameters and the KPIs relevant to given parameters. This extends the scope of data required for ConMit operation that needs to be setup by the operator. The proposed solution also considers assignment of weights to the xApps by a dedicated conflict supervisor xApp to align the ConMit activities with the current policy configured by the MNO.

While CMF itself does not require xApps to provide any non-standard information, QACM introduces a requirement on xApps to provide optimal ranges of control parameters upon request. Furthermore, QACM requires that xApps are able to provide predictions of relevant KPIs to the central ConMit controller. These interactions with a ConMit component make sense in the solution proposed by Wadud et al. as they enable the conflict resolution logic in QACM. However, as these interactions are not currently envisioned in O-RAN specifications, it would be impossible to successfully apply QACM in an O-RAN with COTS xApps.

It must be mentioned that it is possible that future revisions of O-RAN specifications are extended with relevant extensions to accommodate requirements of QACM. Even with such extensions being in place, the considered solution may suffer from biased input data. Specifically, since CR logic of QACM utilizes KPI predictions provided by xApps, there is a risk that vendors will manipulate the predictions provided by their xApp to cause QACM (or any similarly functioning ConMit solution) to prioritize decisions made by the xApp. The incentive for an xApp vendor to introduce such bias would be to ensure that their xApp meets its target objectives, even at the cost of other xApps not meeting theirs.

Operation of QACM requires that, upon detection of a conflict, the ConMit component communicates with each conflicting xApp to receive information about optimal configuration ranges for conflicting control parameters. In parallel, QACM may request a dedicated conflict supervision xApp for allocation of weights to xApps in accordance to the current MNO policy. All of this

extraneous communication takes time and, depending on implementation of ConMit and the xApps, may result in significantly delayed control decisions.

Performance of QACM was assessed by Wadud et al. in an environment with four xApps: CCO, Energy Saving (ES), MRO, and MLB, and contrasted against game theoretic algorithms NSWF and EG [22]. Further assessment in [23] included comparison against prioritization and set-back-to-default methods (later of which works by setting the parameter to default value upon conflict detection) in a scenario with conflict between ES and MRO functions. In both of these comparisons, QACM showed great performance in comparison to the baseline algorithms leading to xApps being closer to reaching their optimization goals despite ConMit activities.

COMIX in CMF

COMIX, or generalized Conflict Management scheme for Multi-Channel Power Control in O-RAN xApps, is a rule-based ConMit method based on CMF described by Giannopoulos et al. [19], which is designed to resolve conflicts between applications controlling transmission power of cells; as such, this is a case of direct conflict. COMIX utilizes a digital twin of theO-RAN (a simulated digital representation of the network) for the assessment of impact of specific control decisions. Based on one of the defined CR policies, KPI estimates from the digital twin are processed into a single scalar score value used to select the applied control decision; other conflicting decisions are rejected. As such, the CR method in COMIX can be considered as a dynamic policy-based prioritization method without cooldown.

The power-controlling xApps considered in the evaluation scenario for COMIX are EE and Data Rate Maximization (DRM). The former targets to optimize energy efficiency via maximization of data rate to power consumption, while the latter focuses on adapting transmission power of cells to maximize the total data rate. The five CR policies described in the article introducing COMIX are based on the characteristics relevant to EE and DRM xApps. Three of these policies are relatively simple and target the following optimization goals: maximize total throughput, minimize total power consumption, maximize ratio of throughput to power. The remaining two policies, one focusing on energy efficiency and the other on throughput, differ from the others by considering relevant UE-level QoS violations instead of network-level metrics. This QoS-aware approach is similar to the CR solution introduced in QACM [22]. These policies assess the negative impact of control decisions by summing up violations of the considered QoS threshold that would result by executing the decision. In case of conflicting decisions with equal validation-based assessment score, the policy favors the decision with the lowest power setting.

While the design of COMIX enables robust CR operation, its dependency on network simulation via a digital twin is problematic due to several reasons. First, efficiency of the solution is highly dependent on the quality of the digital twin and its correlation with the real network. In corner cases, discrepancies between the simulated and actual environments may cause ConMit to make suboptimal decisions, negatively impacting performance or reliability. Second, computing the digital twin's feedback regarding impact of each control decision can take significant time. The required computing time depends the twin's implementation and the network's complexity. Since each conflict requires at least two executions of the simulation, COMIX is likely to cause significant delays during its CR operation. Another concern regarding this solution is that it assumes that the ML-based xApps deployed in the network are trained using the same simulated environment as the one used in the operation of COMIX. While this is a valid assumption for the research phase, it restricts commercial deployment possibilities to MNOs that maintain continually updated digital twin representations of their networks with statistical data reflecting current conditions.

Evaluation of the proposed policies was conducted in comparison to a case without ConMit. In all scenarios, enabling resolution of conflicts with COMIX substantially reduces network-level power consumption. Only the policy prioritizing maximum total throughput of the network yields a small increase in network data rate, albeit with the smallest improvement in power consumption over the baseline. The remaining four policies achieve similar power savings but result in small reductions in total throughput of the network. Overall, it can be concluded that introduction of CR with COMIX improves certain KPIs while causing only minimal declines in others.

PACIFISTA

PACIFISTA is a ConMit solution for O-RAN described by del Prever et al. [21]. It is deployed in SMO and it covers detection, evaluation, and mitigation of conflicts, accounting for control decisions from rApps, xApps and dApps. PACIFISTA differs from the ConMit solutions described above: its ConMit operation relies on application profiles compiled based on the set of parameters modified by given app and the statistical information about the app's behavior. The gathered statistics include probabilistic distributions of control decisions made by each application under specific conditions (i.e., number of UEs, cell load, channel quality, etc.).

Based on the compiled application profiles, PACIFISTA is able to detect conflicts by building parameter and KPM graphs. The parameter graphs models relations between control parameters, while the KPM graphs model relations between control parameters and KPMs. The detected conflicts are categorized into three groups:

- direct conflicts - multiple apps modifying the same parameter,
- parameter conflicts - multiple apps modifying interdependent parameters,
- KPM conflicts - multiple apps modifying parameters, which influence interdependent KPMs.

Authors of PACIFISTA consider indirect and implicit conflicts as effectively equivalent, since implicit conflicts can be detected pre-action with ML solutions (e.g., such as presented in [142], which will be discussed later in this section). Consequently, indirect and implicit conflicts are covered in parameter and KPM conflicts.

Another unique aspect of PACIFISTA is that its ConMit operation includes evaluation of conflicts with regard to their severity. This step is conducted upon detection of a conflict. Severity of conflicts is assessed based on the statistical distributions computed for applications; the considered metrics include Kolmogorov-Smirnov (K-S) distance (maximum vertical distance between distribution functions) and integral of the absolute distance between distribution functions. These metrics are computed into conflict severity indicators by aggregating weighted distances between distribution functions of the conflicting applications. Upon detection and assessment of a conflict, PACIFISTA is able to generate a report detailing all conflicting applications, the scope of the conflict (parameters, KPMs), the network conditions during the conflict, the relevant parameter and KPM graphs, and the computed severity.

Mitigation of conflicts in PACIFISTA operates at a relatively high level. Specifically, it does not interfere with specific conflicting decisions by modifying or rejecting them, but instead it may choose not to deploy some of the conflicting applications. As MNOs are likely to accept low-severity conflicts, a severity threshold is configured within PACIFISTA to allow such tolerance. Furthermore, the operator must assign priority indexes to all deployed applications. The decision to deploy an application is conditioned by its priority and the severity of detected conflicts related to that application. Applications are considered for deployment in descending priority order and are

deployed only if the severity of their conflicts with the already deployed apps does not exceed the configured threshold. Since conflicts detected by PACIFISTA are context-specific (i.e., they may occur under some conditions but not others), the set of deployed applications must be reconsidered whenever network conditions change.

The simplicity of PACIFISTA's CR action space has potential downsides. Effectively, its approach is binary in context of a single application, as the decision is whether to deploy the app or not. This reduces the flexibility of the ConMit in comparison to solutions which mitigate conflicts at the control decision level. Some applications implement multiple optimizations across different use cases; in case of a conflict related to a portion of an application's optimization activities, PACIFISTA may choose not to deploy the application in its entirety. Consequently, even relatively minor conflicts in specific optimization tasks can lead to the complete exclusion of such application, potentially discarding its valuable, non-conflicting functionality.

PACIFISTA was trialed using the Colosseum network emulator [81, 82] and set of eight xApps, including five stochastic ones for slicing (slice-level PRB distribution) and three DRL-based ones for slicing and/or scheduling (assigning PRBs to UEs). The experimental results show step-by-step operation of PACIFISTA and provide a detailed analysis of the impact of conflicts on the network KPMs.

Solutions for ML-based conflict resolution

Utilization of AI/ML techniques is one of the promising direction in research of CR solutions due to their ability to learn optimal behavior policies based on past observations. This characteristic of AI/ML algorithms is important specifically in the context of O-RAN, where multiple applications coexist, and the setup of deployed applications changes over time. A well designed and trained CR method using such techniques is expected to be able to adapt to dynamically changing network conditions without deterministically defining the logic for such adaptation. Depending on the design and scope of the CR solution, it may adapt to addition, removal and reconfiguration of applications.

ConMit solutions using AI/ML techniques may utilize the same action spaces as conventional rule-based methods. As an example, **Cinemre and Mahmoodi [24]** proposed a ConMit method using the RL A2C algorithm for scheduling activation of specific xApps deployed in the network according to the current conditions and the MNO's policy. This mode of operation can be compared to how PACIFISTA [21] performs its ConMit actions. The difference is that PACIFISTA uses the O1 interface to deploy chosen xApps (and dApps; other than that it uses internal messaging within SMO to deploy rApps), while the proposed scheduler uses a proprietary activation message to the inference host of xApps to indicate which applications to enable. These two approaches are effectively the same in relation to impact on operation of xApps.

Cinemre and Mahmoodi consider the scheduler-based ConMit solution as a part of a larger intent-driven framework for deployment of applications. The framework envisions two dedicated logical entities deployed in the SMO for the sake of intelligent orchestration:

- Intent-Based Network Management (IBNM) - acts as an interface for MNOs to declare their intents via high-level language and converts these intents into low-level constraints using an LLM,
- Intelligent Orchestration (IO) - monitors deployed application to ensure that they work in alignment with the current intents provided by the MNO and prevents conflicts by restricting multiple applications to influence the same control parameters.

When the operator specifies an intent for network optimization via IBNM, the Non-RT RIC selects the target applications to deploy to fulfill the intent; the exact logic for how IBNM, IO and Non-RT RIC should execute these tasks is not defined in the work by Cinemre and Mahmoodi. The intent along with additional context variables are provided into the Near-RT RIC via the A1 interface's EI. There, the proposed ML-based scheduler ingests the information to decide which xApps to activate.

The proposed solution is evaluated using a simulated scenario with power and PRB allocation xApps. These xApps are in indirect conflict, as both impact throughput of UEs. The considered ML-based xApps use A2C algorithm for their logic and use total data rate as reward during training. In the study paper the scheduler is limited to optimize only a single metric, specifically to maximize the transmission rate, similarly as the ML-based xApps. This optimization target coincides with the operator's intent in the evaluation scenario. The context variables provided by SMO via A1 are average UE movement speed and average data arrival rate. These context variables are used as input for the scheduler's RL algorithm.

As the proposed approach with the scheduler has a similar action space to PACIFISTA, it also has the same downside regarding its inferior ConMit flexibility due to disabling entire applications. Furthermore, since the proposed scheduler relies on a proprietary interface to communicate with the xApp inference host and activate relevant applications, implementation of the scheduler requires extensions outside of the O-RAN standard.

The authors consider two operation modes of the scheduler. In the first one, only the two ML-based xApps are considered and the scheduler whether decides to activate them. In the other one, the ML-based applications have their baseline rule-based alternatives; the scheduler then selects which xApp (ML- or rule-based) to activate for each of the two optimization goals. The results of the evaluation are presented by the authors in three variants of context variables (low, mid, high), the results are varying more in the case of higher values of context variables. It can be observed that the proposed scheduler approach improves significantly on the case with both deployed xApps without any ConMit measures. However, the case of the scheduler working in the mode with only ML-based xApps does not provide better results than single deployed applications. Introduction of the baseline xApps working as alternative to the ML-based ones enables the best performance across the tested variants.

Zafar et al. [143] present a ConMit approach powered by Double DQN (DDQN), tailored for the conflicts between EE and MLB xApps, which operate by disabling (performed by EE) and enabling (performed by MLB) O-RUs. The proposed ConMit algorithm ingests utility functions from the xApps and takes the action to disable or activate each O-RU. The reward function used for the training of the ConMit DDQN combines a weighted sum of utility functions of the considered xApps with a factor which takes a negative value in case feasibility constrains of xApps are not met, and positive value otherwise.

There are several problems related to the approach proposed by Zafar et. al. The considered xApps and the proposed ConMit scheme operate by disabling and/or enabling O-RUs and the described logic applies only in this specific case. This makes applying the proposed solution to any other xApp deployment scenario challenging at the least, and potentially impossible in some cases. Furthermore, the scheme requires that xApps are able to provide values of their relevant utility functions to the ConMit component; such interface is not envisioned in the O-RAN standards, so the proposed scheme requires implementation of some proprietary extensions, hindering interoperability.

The results of simulation for evaluation of the proposed ConMit scheme include comparisons against individual xApps operating independently and a brute force joint optimization approach that exhaustively searches the action space to find the optimal resolutions. The results show that the proposed solution outperforms the individual applications. Its performance converges towards the optimal results of the brute force approach, but does not reach the same level of performance. The authors argue that the DDQN-based ConMit approach reaches a good trade-off between performance and computational complexity required to compute a solution.

ConMit methods utilizing AI/ML techniques are capable of CR strategies more indirect than modifying conflicting control decisions or disabling/enabling xApps/O-RUs. An article by **Erdol et al.** [**20**] describes a method for distillation of policies of multiple xApps into a single application. This solution works by collecting past experiences of source xApps and using them to train the target application powered by a DQN. This approach is described by Erdol et al. as suitable for both ML- and rule-based applications.

The implemented DQN's action space must include all actions that the source xApps are capable of. In the article, the proposed ConMit solution is tasked with taking over the responsibilities of two xApps; one optimizes handovers and PRB allocations, while the other optimizes handovers and cell transmission power. As such, the implemented ANN is a multi-headed DQN capable of executing handovers between cells, allocating PRBs to UEs, and controlling transmission power of cells. The inputs of the proposed ANN include UE's SINR values for nearby BSes, a value of utility function as logarithmic representation of scaled downlink data rate, and a one-hot encoded array indicating the UE's connected BS. The reward for the DQN is calculated by summing up downlink data rates for a set of users in the logarithmic scale; using the logarithmic scale acts as a way to prioritize fairness as it penalizes extremely low and high data rates.

Policy distillation, which is the basis of the CR solution proposed by Erdol et al., aims to transfer policies from multiple sources (in this case, source DQNs, but this method applies to any applications that produce an action based on observed state) into a single target (target DQN). The authors envision this approach as applicable in the case where an MNO purchases several applications from various vendors and deploys them in their O-RAN environment. Initially, the purchased xApps come as pretrained by the vendor in their environment; the MNO is not likely to have insight into the internal intricacies of any 3rd party application. Policy distillation requires that the source applications are first deployed in a target environment (production environment or testbed), so that an experience memory of the actions taken by each source application can be captured. To avoid influence of any existing ConMit mechanisms present in the environment, the data is saved before the action is applied to the network by the xApp. Once all of the required data are collected, the info stored in the replay buffer is used to train the target DQN, which is then deployed in the environment for testing and evaluation.

Despite its appeal, the proposed distillation approach generates significant hurdles for the operator. First, due to the need to compile experiences from many source applications in a format suitable for the target ANN, significant data processing is likely to be required to translate the source input formatting into target input formatting. Furthermore, adding new applications to an environment with existing application with distilled policy requires rerunning the entire distillation process with inclusion of the new xApp. These two caveats mean that the proposed distillation approach is not a "plug & play" solution and will increase maintenance costs for implementation of data processing for the MNO.

While Erdol et al. claim that indirect conflicts are "almost impossible to observe prior to the action" [20], they effectively conflate indirect with implicit conflicts. As demonstrated in Section 5.4.3, indirect conflicts can be detected pre-action using parameter groups.

The evaluation of the solution is performed against a case with no ConMit and a case with applications with TL as described in [90] (described in detail below). As mentioned, the authors consider a case with two conflicting xApps: one optimizes handovers and PRB allocations, while the other optimizes handovers and cell transmission power. The authors present probability density functions for downlink data rate and network outage percentages in relation to target throughput in all evaluated cases. The evaluation results show that the proposed policy distillation approach outperforms both no-ConMit and team learning approaches in both data rate and network outage metrics.

As a somewhat of an honorable mention in context of existing CR approaches, the work by **Zhang et al. [90]** can be mentioned. This article was already described in Section 2.4.4; it presents a TL-based approach for xApp coordination, in which participating applications are aware of each other's control decisions and take them into account in their own decision-making logic. This approach can be considered as a ConMit method, but, as mentioned earlier, it puts significant requirements on the xApps to be able to consider decisions of other applications. As such, this solution is not likely to be applicable in many O-RAN deployments.

Solutions for conflict detection

Methods for conflict detection are not as explored in terms of research as CR methods, but, at the same time, there are not as many gaps in this field. The detection of direct and indirect conflicts is relatively simple and its efficiency can be reliably determined. On the other hand, detection of implicit conflicts is subjective and various methods may be hard to assess and compare in a fair manner (as already discussed in Section 3.2). As such, in terms of efficiency of CD methods, the main challenge for the industry is detection of implicit conflicts.

Basic rule-based CD approaches for direct and indirect conflicts were first described in the **article written by the author, introducing CMF [2]**. Both these conflict types are detected in CMF pre-action with simple logic. Indirect conflicts require groups of related parameters to be configured in the RIC, while detection of implicit conflicts is conducted post-action as it relies on tracking correlation of performance degradation occurrences with past control decisions. Details of CD methods introduced as part of CMF are further detailed in Section 5.4.3.

Another variant of the CMF rule-based approach for CD is presented in works by **Wadud et al. [18, 23]**. It utilizes similar scope of data related to parameter groups, recently changed parameters, and performance degradation occurrences. However, proposed method for CD works post-action for all conflict types (and not pre-action for direct and indirect conflicts as in CMF). Specifically, the conflict is detected only after detecting performance degradation. Then, the proposed algorithm considers recently changed parameter and parameter groups to determine the type of detected conflict. Such approach introduces a significant delay in executing ConMit action for direct and indirect conflicts as the conflict must first influence a monitored KPI enough to trigger the CD mechanism. Furthermore, this method may be insensitive to performance fluctuations which may not be directly visible to simple monitoring algorithms based on thresholds. These

fluctuations are undesirable from the perspective of an operator, even though they do not lead to significant performance degradations.

Armstrong et al. [144] propose a clustering-based CD approach, which seemingly targets conflicts between rApps deployed in the Non-RT RIC. Although the article's title in context of O-RAN may hint at it describing a method for detection of conflicts between rApps, it is rather a method for detection of configuration changes resulting in negative performance impacts. The proposed method works by monitoring and comparing temporal behavior of KPIs in cells working in similar conditions. Clustering is used here as a method to categorize the cells into groups based on specific configuration parameters. Then in case of a change in parameters of a cell provided by any of the rApps, similarity between KPI dynamics of specific cells within a group is calculated using a Pearson-correlation-based algorithm to detect the most similar cells in the group with the initial parameter value and the target parameter value. The CD mechanism then assesses whether accepting the new parameter value will increase or decrease KPIs of the cell. In case of a detected performance deterioration, the rApp that provided the control decision is informed; this method requires any supported rApps to be able to ingest such feedback.

Usage of graphs is one of the key trends in conflict detection in O-RAN. There are already many various CD approaches utilizing graph-based conflict representations. One of such methods is the already described **PACIFISTA solution [21]**, which performs its CD by analyzing parameter and KPM graphs. The graphs in PACIFISTA are built with three node types representing applications, parameters, and KPIs; directed vertices between nodes represent influence. As such, in PACIFISTA, these graphs act as mathematical representations of relations between applications, parameters, and KPIs.

Other graph-based solutions utilize GNNs as a method to process data in graph form. Specifically, **Al Shami et al. [145]** describe a CD solution named Graph-based xApps Conflict and Root Cause Analysis Engine (GRACE) that uses a Graph Convolutional Network (GCN). Similarly to PACIFISTA, the graphs in GRACE model relations between applications, parameters, and KPIs. This solution executes three main tasks: dataset generation, creation of graphs, and detection of conflicts based on anomalies in the created graphs. First, binary-state dataset is computed from network observations. This dataset includes binary indicators for each parameter about which applications modify it, and for each KPI about which parameters influence it. Another indicator is maintained to hold information about pairs of parameters which influence each other. Subgraphs are built based on the collected datasets. Then, the subgraphs are combined into a single target graph for further processing. This graph is used as input to a two-layer GCN, which learns node-level representations, which are then aggregated via pooling into a graph-level representation finally used for analysis via a fully connected neural network layer. Output of this analysis includes one of four classifications: no conflict, direct conflict, indirect conflict, or implicit conflict. To evaluate GRACE, five binary-state datasets were generated with data having various distributions of all conflict types (including no conflict). Authors showcase the method's efficiency, which reaches over 98% prediction accuracy rating. Furthermore, the graph representations created as part of the data analysis allows to determine which xApps contribute to detected conflicts, allowing for a root cause analysis.

A different CD approach using GNNs is proposed by **Zolghadr et al. [142]**, further detailed in Zolghadr's PhD thesis [146]. Unlike GRACE, which employs a relatively simple GCN to infer conflict classes from binary-state observations, the approach proposed by Zolghadr et al. utilizes the GraphSAGE framework, tailored for inductive learning on large graphs, to learn and reconstruct conflict graphs. GraphSAGE is based on the concept of GNN, but it applies sampled neighbor aggregation instead of full-graph convolutions; this enables training with mini-batches and generalization to unseen nodes and unknown graphs. The proposed method incorporates temporal graphs to capture correlations between observations in subsequent time steps. These temporal graphs are then used as input to the GraphSAGE ANN, which learns embeddings that represent graph nodes and outline the correlations between specific nodes. Weak links are filtered out with an empirically chosen threshold, and the refined data is combined with knowledge of application-parameter relations and application-KPI dependencies. This enables the proposed algorithm to detect conflicts by deriving implicit relations between parameters and KPIs and constructing a complete conflict graph. As a result, the proposed ML-based solution is capable of detecting implicit conflicts. Evaluation of the proposed scheme shows that, with enough data points and training epochs, detection accuracy of 100% can be reached.

Sharma et al. [147] present an approach for graph-based CD with utilization of ML techniques. Their proposed conflict management solution is able to derive dependencies between control parameters and KPIs based on observations of configured parameter values and respective metrics. Other than detection of conflicts, this solution is able to estimate influence of each control parameter on each KPI to assess impact of detected conflicts on network performance metrics. The implemented regression model is trained to predict KPIs based on control parameter values. Then, a model explainability method Shapley Additive Explanations (SHAP) is used to analyze which control parameters influence which KPIs. A Directed Acyclic Graph (DAG) is constructed based on the SHAP analysis to reflect causal relations between parameters and KPIs, enabling detection of indirect and implicit conflicts. But such correlation-level dependencies are not enough to explain the conflicts completely. Each edge of the computed DAG is weighed by calculating two metrics describing impact of parameter on KPI: Average Treatment Effect (ATE) and Conditional ATE (CATE). The former describes average change of metric caused by increase of parameter by one unit, while the latter provides an extension of considering current network conditions. Other than conflict detection, this method provides additional insights into the nature of conflicts and may prove helpful in evaluating CR solutions and monitoring ConMit efficiency in runtime. As the framework abstracts away from considering conflicts in context of xApps, additional processing in ConMit component is required to link specific controlled parameters to deployed xApps. The proposed solution was evaluated through computer simulation. Five ML models were tested for regression as part of the paper, with XGBoost identified as the best performer over models such as Multilayer Perceptron (MLP) and Support Vector Regression (SVR); this model was used for further evaluation of the solution. Analysis with SHAP showed that changes in transmission power have the largest influence on throughput and Block Error Rate (BLER), while spectral efficiency is impacted the most by bandwidth. Based on the presented SHAP analysis results, a DAG is constructed and causal metrics (ATE and CATE) are calculated. Obtained values of ATE show several indirect conflicts between considered parameters. Example of such conflict is transmission power and bandwidth both significantly influencing spectral efficiency. Presented

evaluation scenario does not show the case of detection of implicit conflicts, hence it is not clear how efficient this solution would be for this purpose.

Usage of graph-based solutions for detection of conflicts requires frequent gathering and processing of data. As such, relying on these approaches to detect direct and indirect conflicts seems unnecessarily complex. However, these methods are well-suited to detect implicit conflicts, as they are able to learn and communicate the relations between parameters and KPIs.

Other related works

Sultana et al. [148] describe an Software-defined Radio (SDR)-based platform capable of OTA transmission for detecting and resolving conflicts in O-RAN. The design of their platform is in line with principles of CMF. The main showcase of the platform is a case of direct conflict between xApps, where basic first-come first-served logic is used to mitigate the conflict. An evolution of this platform was later used in hardware-based evaluation of CMF [4], as described in details in Chapter 7.

A game-theory-based CR approach is proposed by **Wadud et al. [18]**; it allows to mitigate conflicts of all three types. Similarly to QACM method in other work by Wadud et al. [22], it is based on CMF. It expands on the requirements of CMF, as it requires xApps to provide its priority weight (assigned by MNO), parameter ranges and KPI predictions as demanded by the ConMit component of Near-RT RIC and react to parameter value guidance from the ConMit component provided after CR computing is completed. The proposed ConMit scheme includes a procedure, where xApps provide predicted KPI values upon reception of the initial parameter value guidance. The ConMit component then considers these predictions to calculate values of utility functions for specific xApps and determine the final optimal parameter value suggestion, which is then communicated to the application and applied in the network. Computation of the optimal parameter value done as part of the algorithm proposed by Wadud et al. is conducted using NSWF or the EG solution to derive an optimum for provided priority weights, parameter ranges and KPI predictions along with calculated values of relevant utility functions. Evaluation of the proposed solution done by the authors shows that it is indeed capable of deriving the target optimal value of parameters based on the information exchange with conflicting applications.

Such complex exchange information procedure, as proposed by Wadud et al., introduces potentially large delays before a conflict gets resolved, leading to negative impacts of conflicts lingering for longer than needed. Furthermore, it introduces a number of non-standard requirements for xApps, which need to support the proposed ConMit approach, making this solution not likely to be widely supported by 3rd party applications. Moreover, results of the operation of this scheme are highly dependent on formulation of the utility functions assigned to each xApp.

An interesting solution in context of O-RAN ConMit is described by **Hou et al. [149]** – their work touches on the topic of what-if analysis in O-RAN. Specifically, they developed a method named Counterfactual Conformal KPI Estimation (CCKE), which aims to estimate values of counterfactual KPIs, i.e., the values that specific KPIs would have assumed had different control

decisions been made in the environment. Such capability is important in any case where one xApp's observations are influenced by another xApp's control decisions. Similarly, this capability may become very helpful in logic of ConMit methods to assess impacts of considered control decisions and their modifications as part of resolution process. Additionally, CCKE may act as a way to assess ConMit methods, enabling estimation of KPI values with and without ConMit activities in the same testing run.

CCKE is a statistical framework designed for wireless networks, which outputs counterfactual KPI estimates along with reliable error ranges. Its processing starts with collecting a data set containing context information, information about selected app, and current KPIs values. Then, a controller hosting CCKE targeting estimation of KPI values in given context with a different application than logged in the data set extracts a portion of data from the total available data set that relies to the target application and uses this information to train a predictor and calibrate the predicted error range that shall contain the true KPI value. The calculations use Weighted Conformal Prediction (WCP) as a way to build on basic Conformal KPI Estimation (CKE) by calibrating prediction intervals for KPI estimation.

Evaluation of CCKE is done in comparison to two baseline methods: CKE and Naïve CCKE (NCCKE) (which ignores one of the conditions to guarantee reliability of the predicted KPI values). The results show that CCKE outperforms the other methods and always meets or exceeds target coverage performance regardless of number of users in the testing scenario and value of control parameter considered by the application. Application of the proposed scheme in real networks (e.g., as part of larger ConMit solutions) relies heavily on its precision, i.e., the width of the predicted error ranges that contain the true values.

A curious take on mitigation of conflicts in 6G O-RAN is presented by **Corici et al. [150]**. The authors notice that xApps are almost always developed and evaluated in standalone conditions, not taking into account impacts on other active optimization processes, and propose a ConMit approach based on a classification of xApp use cases. Each application is assigned goal and management functionality classifications. Goals of xApps include the following: performance (P), quality (Q), robustness (R), security (S), energy efficiency and others (+). Possible management functionalities are related either to topology (T) or users (u), or both. The primary part of the coordination between various applications is done by the Topology Management (TM) scheduler, which defines the parameters of system components. Then, user-level optimization (handovers, resource allocation) is conducted by User Management (UM) entity according to commands from TM and based in context of the temporary topology configuration done by TM. Simultaneously, UM communicates to TM events related to user traffic, such as cell overloads and periods of traffic silence; these events trigger the TM component to apply appropriate modifications to the topology. Near real-time actions of UM must not collide with the real-time radio resource control functionality managed by the Service Continuity (SC) component. SC additionally provides information about its operation to UM, so that UM can optimize parameters with accurate knowledge about the system.

Although the presented xApp classification and the resulting framework looks promising and may yield good results, the authors do not present algorithmic and/or procedural details of how this framework could work in a specific scenario. In addition, no results of evaluation of the proposed solution is shown. As such, it is problematic to assess the efficiency of this framework.

3.5 Comparison of O-RAN Conflict Mitigation solutions

Table 3.3 presents a comprehensive comparison of research solutions for O-RAN conflict mitigation. The table details the specific problems addressed, the types of conflicts considered, the specific applications used in evaluation, and the key findings and limitations of each approach.

Table 3.3: Comparison of Conflict Mitigation solutions for O-RAN

Author	**Ref**	**Problem/Challenge**	**Conflict Type**	**Evaluation Apps**	**Findings**	**Features**	**Limitations**
Adamczyk & Kliks (Author)	[2, 5]	Comprehensive framework for CD and CR	All types	MRO, MLB	Prioritization with or without cooldown is effective for indirect conflicts	CMF architecture; rule-based resolution	Simple logic (prioritization)
Wadud et al. (QACM)	[22]	Ensuring QoS targets during resolution	All types	CCO, ES, MRO, MLB	Outperforms baseline game theoretic algorithms; closer to optimization goals for apps	Optimization of parameter values; Z-score utility functions	High complexity; Requires non-standard xApp data (predictions)
Giannopoulos et al. (COMIX)	[19]	Power control conflicts	Direct	EE, DRM	Reduces power consumption with minimal throughput loss	Digital Twin-based; 5 QoS-aware policies	High latency (simulation time); relies on Digital Twin accuracy
del Prever et al. (PACIFISTA)	[21]	Conflict management via app deployment	All types	Slicing, Scheduling	Effective high-level mitigation; Identifies conflict severity	Graph-based CD; Binary mitigation (deploy/don't deploy)	Inflexible app impacts (on/off); discards non-conflicting app functions
Cinemre & Mahmoodi	[24]	Dynamic activation of xApps	All types	Power, PRB alloc.	Scheduler improves over no-CM; Baseline+ML mix works best	RL (A2C) Scheduler; intent-driven (IBN-M/IO)	Binary action space; Requires proprietary interface
Zafar et al.	[143]	RU management optimization	Direct (RU state)	EE, MLB	Outperforms individual xApps; nears brute-force optimality	DDQN-based; Action space: enable/disable O-RUs	Specific to RU on/off logic; non-standard utility exchange
Erdol et al.	[20]	Distillation of multiple xApps into one	All types	HO+PRB vs HO+TxP	Outperforms Team Learning and No-CM cases	Policy Distillation (DQN); single target app	High maintenance (retraining); Not "plug & play"
Al Shami et al. (GRACE)	[145]	CD via GCN	All types	generic	Achieves ¿98% detection accuracy	GCN; binary-state datasets; Graph-based	Limited to detection; requires graph construction
Zolghadr et al.	[142]	Detection of all conflicts	All types	Generic	Can reach 100% accuracy with sufficient training	GraphSAGE; temporal graphs; Inductive learning	Limited to detection; training data dependency
Sharma et al.	[147]	CD and impact evaluation	All types	TxPower, Bandwidth	SHAP analysis effectively identifies indirect conflicts	XGBoost + SHAP; causal metrics (ATE/-CATE)	Abstracts away from xApps; no implicit detection shown in eval

Chapter 4

Implemented simulation environment

A proprietary simulation environment was implemented to enable efficient evaluation of proposed ConMit solutions. Python was chosen as the coding language, as it supports object-oriented programming suitable for large-scale simulations. Furthermore, it offers a wide range of libraries, including ones enabling implementation of AI/ML algorithms. One of the benefits of using Python is its scripting functionalities, making file-based data and log computing possible within the simulation program and streamlining the data gathering process as a result. Version 3.9.2 of Python was used in the final iteration of the simulator.

The development of a dedicated high level Python-based simulator was motivated by the lack of available computation-efficient O-RAN simulators with robust possibilities to implement custom network steering logic, while also enabling highly automated data gathering and processing. For example, the ns-3 is an open source simulator [151] written in C++ language, designed for packet-level network simulation. Through the ns-O-RAN extension [152], it supports integration with real-world O-RAN components. These characteristics also apply to OMNeT++ simulator [153]. Although both of these simulators include partial support for O-RAN experimentation, they rely on integration with fully implemented RICs. Consequently, developing proprietary extensions, like mechanisms for ConMit, require implementing complex C++ modules within both the simulator and the RIC. Such an approach introduces considerable development overhead, which is impractical within the scope and time constraints of this doctoral work.

Considering the aforementioned limitations of existing simulators, the decision to develop the high-level proprietary simulator was made. It is designed to model RAN behavior at an appropriate level of abstraction to capture relevant characteristics of the network while maintaining low computational and implementation complexity. This approach enables efficient experimentation during the research process, with the simulator acting as a practical and extensible platform for implementing and evaluating ConMit solutions.

4.1 Project structure and libraries

The project of the implemented simulator is split into multiple Python modules, which can perform any or both of the following two categories:

- configurational - modules with parameters of the simulation and network,
- functional - modules with object and method definitions used to handle the simulation, manage ML, model the network, and process logging and result files.

The module split of the simulator was developed as the simulator was evolving into its current state. Grouping related functionalities into a single module allowed simple navigation within the code and, as a result, made development and troubleshooting easier. The complete list of modules is shown in Table 4.1.

Module name	Roles	Description
config.py	Configurational	General configuration of simulator, network, and CM measures
cqi.py	Configurational	Configuration of CQIs
ml_params.py	Configurational	Configuration of ML training processes and ANNs parameters
user_profiles.py	Configurational	Configuration of user traffic profiles
gui.py	Functional	Implements GUI of the simulator
main.py	Functional	Implements startup logic of the simulator
ml.py	Functional	Implements ML training processes and ANNs
objects.py	Functional	Implements logic for each element in simulated O-RAN network (cell, user, RIC, xApps, etc.)
prop_model.py	Functional	Implements propagation model for path loss calculations
simulation.py	Functional	Implements logic of the main simulation loop
utility.py	Functional	Implements utility functions reused across many modules

Table 4.1: Simulator project structure

Source: own elaboration

From implementation of ANNs and their training to GUI, the simulator uses a wide range of diverse Python packages to significantly reduce implementation efforts. The list of packages used in the simulator project are listed in Table 4.2. Please note that built-in Python packages do not have a version specified.

Package name	Version	Utility
Built-in Python packages		
argparse	N/A	Parse command-line arguments and options
bisect	N/A	Search for index in simulator event list when scheduling new events
csv	N/A	Read configuration of simulation runs, write simulation results
datetime	N/A	Human-readable simulation duration formatting in GUI mode
enum	N/A	Define enumerations for RIC logic
json	N/A	Encode data in JSON format for logging
logging	N/A	Record logs and events during simulation
math	N/A	Perform mathematical operations for network modeling
os	N/A	Create directories, generate file paths
random	N/A	Generate random numbers and selections
shutil	N/A	Move log files to dedicated directory after each simulation run
sys	N/A	Access system-specific parameters and functions
threading	N/A	Run multiple threads concurrently to separate simulation loop from main program and GUI loops
time	N/A	Generate timestamps, measure code execution time
traceback	N/A	Print Python stack traces for debugging
Third-party packages		
matplotlib	3.5.1	Plotting and 2D visualization of data in GUI mode
numpy	1.22.1	Numerical computing of metrics and statistics for network modeling
torch	1.12.1+cu113	Deep learning framework for neural networks
wxPython	4.1.1	GUI mode of the simulator

Table 4.2: Packages used in the simulator

Source: own elaboration

4.2 Simulation flow

The simulator is based on discrete events. All activities possible during the simulation are defined as events, including movement of UEs, establishing connections between UEs and cells, performing UE handovers between cells, logging simulation statistics, etc. Events are managed in

a list, which is ordered sequentially by each event's execution time. This list ("event list") holds all events, which are yet to happen during the simulation. Another list ("history list") holds the events, which have already happened. Each instance of simulation runs until the target simulation time is reached. Upon conclusion of the simulation run, the simulator saves the captured statistics to files and becomes available for further operation.

4.3 Network model and events

The O-RAN network modeled in the simulator consists of various components interacting with each other. The component classes comprising the network model implemented in the simulator are listed in Table 4.3. Descriptions of how these components interact with each other are provided in the following paragraphs.

Component class	Description
	UEs & connectivity
`user_profile`	Defines the traffic profile of a user (UE)
`user`	Represents an active user
`unique_user`	A unique user instance used for troubleshooting
`connection`	Models connections between users and cells
	Network Infrastructure (BS & Cell)
`base_station`	Base class for the radio base station
`lte_bs`	Represents a 4G LTE base station
`nr_bs`	Represents a 5G NR base station
`cell`	Base class for the radio cell
`lte_cell`	Represents a 4G LTE cell
`nr_cell`	Represents a 5G NR cell
	Propagation Models
`prop_model`	Base class for pathloss and radio propagation calculations
`COST231_model`	Implements the COST 231 Hata propagation model
`TGPP_38_901_UM_model`	Implements the 3GPP 38.901 Urban Macro and Micro models
`TGPP_38_901_UM_model_sector`	Sector-specific 3GPP model variant, including horizontal/vertical power cut calculations
	O-RAN RIC
`ran_controller`	Near-RT RIC, hosts xApps, PM Agent, and ConMit functionality
`xApp`	Base class for xApps
`mro_xApp`	MRO xApp, optimizes handover parameters based on observed network conditions
`mlb_xApp`	MLB xApp, balances cell load based on observed network conditions
`pm_agent`	PM Agent, triggers processing of all KPIs
`cd_agent`	CD Agent, detects conflicts between xApps as part of CMF
`cr_agent`	CR Agent, resolves conflicts between xApps as part of CMF
`control_conflict`	Data structure representing an instance of control decision conflict
	Simulation Core & Events
`simulation`	Top-level orchestrator, handles simulation setup, runtime control, user management, and result logging
`environment`	Central simulation environment, handles scheduling of events, sets up components, manages user movement, and advances the simulation time
`traffic_generator`	Generates user connections and disconnect events according to a traffic model, associated with a specific user
`event`	Base class for all schedulable actions on the timeline
Various `..._event` classes	Specific events related to network operation (see Table 4.5 for details)
	Performance & Statistics
`global_statistics`	Aggregates and retrieves simulation statistics during each run
`kpi`	Base class for all KPIs
Various `..._kpi` classes	Specific KPI implementations to calculate key metrics for cells and users (see Table 4.4 for details)
`stat_entry`	A single entry for statistical recording

Table 4.3: Components implemented in the simulator

Source: own elaboration

The modeled network can comprise of 4G LTE or 5G NR BSes (or both). All BSes are located in a simulation area constrained by a circle with a radius of R_{area}. Each BS hosts a number of cells providing service in a sector. All cells have their separate detailed configuration describing the radio operation, such as channel bandwidth (B), middle frequency (f_{c}), Subcarrier Spacing (SCS), transmission power (P_{tx}), antenna height (h_{ant}) and gain (G_{ant}), cable loss (L_{cable}), etc.

The simulation area is populated with users belonging to one of the available user throughput profiles assigned at random according to a configured distribution. The users can either be a pedestrian (r_{ped} percent of users moving with v_{ped} speed) or driving in a car (r_{veh} percent of users moving with v_{veh} speed). All users are randomly distributed across the simulation area and cannot cross the circular area boundary when moving. The direction of their movement is random, with a small chance to change at each movement interval and with a guaranteed change at each crossing of the simulation area boundary. Movement of users in the simulation happens periodically for all users at once.

Radio conditions are determined for connections between all users and all cells each time position of users is updated. The path loss for each connection is calculated using one of the implemented propagation models: COST 231 Hata model [154] or 3GPP 38.901 Urban model [155]. These models allow stochastic calculation of radio channel parameters in an urban setting without explicitly modeling the environment (i.e., determining the placement and dimensions of buildings, streets, and so on). Based on the calculated path loss combined with the cable loss, antenna gain, and receiver sensitivity for the user and cell, the received power and SINR for connections are determined.

Each user generates network traffic according to configured traffic model parameters, with the time of the first connection determined for users at the beginning of the simulation. The connection is established with a cell which provides the highest bitrate satisfaction (i.e., the percentage of requested bitrate that the cell can provide). If multiple cells provide the same maximum satisfaction, the user connects to the one with the higher received signal power.

While a user is connected to one of the cells, handover conditions are verified for all other cells every time the user moves. A handover from one cell (source cell) to another (destination cell) is performed when the received power difference is high enough for long enough time – the specific conditions are determined by handover parameters of the cells (CIO, handover hysteresis, TTT). A handover can fail when it is performed too late (connection to source cell fails before connection to destination cell is established) or too early (connection to destination cell fails due to too low channel quality upon handover).

An O-RAN Near-RT RIC is deployed in the network as the main controller responsible for monitoring network performance in addition to hosting xApps and ConMit measures. There are two xApps deployed in the RIC:

- MRO - optimizes the handovers between cells by manipulating handover hysteresis and TTT. Effectively, the optimizations performed by this xApp influence two characteristics of the handover processes in the network: how long the handover conditions need to be met to perform the handover, and how large the difference in a user's received power needs to be between two cells to trigger a handover between them. MRO performs its control actions with the goal of minimizing the number of ping-pong handovers (i.e., concurrent handovers between two cells) and RLFs (i.e., active radio link between cell and UE failing due to too late or too early handovers).

- MLB - also updates the handover parameters, but it manipulates the values of CIO parameter for each cell in the network with the goal to balance the traffic load across the cells. This is

done with the intent to avoid Call Blockages (CBs) – the cases where there is no cell that can provide service to a user trying to connect. Similarly to the handover hysteresis, the CIO parameter is a factor applied to power imbalance condition during a potential handover. However, CIO is used by MLB to promote handovers from highly congested cells to those with a lighter traffic load.

Throughout the simulation process, various statistics and metrics are monitored for specific cells, users, and the entire network. The general simulation statistics regarding each run are handled by the simulation process, while the RAN-specific metrics are gathered by the PM Agent implemented as part of the Near-RT RIC. The PM logic in the RIC is based on cell- and user-level KPIs. The cell- and user-level KPIs monitored by the Near-RT RIC are implemented as dedicated components classes. Their values are periodically calculated for each cell and user, and used for the RAN optimization processes hosted in the RIC. The specific instances of these KPIs are described in Table 4.4.

KPI class	Description	Equation
	Cell-level KPIs	
`cell_availability_kpi`	The percentage of time within the measurement window that the cell load is less than 100% (i.e., the cell is available)	$\text{Availability} = \frac{\text{Count(Times available)}}{\text{Count(Total observations)}}$
`cell_handover_success_kpi`	The average success rate of handovers within the measurement window; calculated as successful handovers divided by attempted handovers	$\text{Success rate} = \frac{\text{HO attempts} - \text{HO fails}}{\text{HO attempts}}$
`cell_handover_stability_kpi`	The average rate of non-ping-pong handovers within the measurement window	$\text{Stability} = \frac{\text{HO total} - \text{HO ping-pong}}{\text{HO total}}$
`cell_throughput_kpi`	The average total throughput aggregated by all connected users served by the cell, where N is the number of measurements in the window and $\text{Throughput}_i^{\text{Cell}}$ is the instantaneous throughput at index i	$\text{Throughput} = \frac{1}{N}\sum_{i=1}^{N} \text{Throughput}_i^{\text{cell}}$
	User-level KPIs	
`user_satisfaction_kpi`	The average satisfaction (ratio of provided to requested bitrate), where N is the number of measurements in the window and $\text{Satisfaction}_i^{\text{User}}$ is the instantaneous satisfaction at index i	$\text{Satisfaction} = \frac{1}{N}\sum_{i=1}^{N} \text{Satisfaction}_i^{\text{user}}$
`user_connection_success_kpi`	The success rate of connection attempts made by the user over the given time period	$\text{Success rate} = \frac{\text{Successful connects}}{\text{Connect attempts}}$
`user_throughput_kpi`	The average throughput achieved by the user's active connection, where N is the number of measurements in the window and $\text{Throughput}_i^{\text{User}}$ is the instantaneous throughput at index i	$\text{Throughput} = \frac{1}{N}\sum_{i=1}^{N} \text{Throughput}_i^{\text{user}}$

TABLE 4.4: KPIs implemented in the simulator

Source: own elaboration

The simulated RIC hosts CD Agent and CR Agent components as part of its implementation of CMF for execution of ConMit. The details of their operation, along with the supported ConMit methods, are described in Section 5.4.2.

Interactions between the components in the network model are all based on discrete events executed in order by the simulator environment. Like the components, the events are implemented as specific classes, each with some shared common event management logic and class-specific execution logic. All event classes implemented in the simulator are listed in Table 4.5.

4.4 Simulation parameters

The specific values for the parameters used in the evaluation scenarios and the generalized network model described in Section 4 are listed in Table 4.6. These parameters are defined in

Event class	Description
event	Base class for all scheduled actions. Handles common scheduling properties (like execution time and event type) and logs the event to the history list of associated objects for given events (e.g., links past event to a relevant cell, connection, kpi, control_conflict, etc.)
	Network events
move_event	Triggers the simulation step responsible for user mobility. Executes movement of UEs, recalculates quality of connections, checks for handover conditions, and schedules the next move_event.
connect_event	Triggers a user to attempt connection to a cell
failed_connect_event	Used exclusively for logging and tracking a failed user connection attempt; does not execute functional logic beyond history tracking
disconnect_event	Triggers a user to disconnect (deactivate an active connection) and schedules a new connect_event via the user's traffic_generator
	Mobility events
handover_check_event	Checks if handover conditions are still met for a pending handover; if the TTT is reached, a handover_event is scheduled. Otherwise, it schedules the next check (if conditions are met) or resets the status (if conditions are not met)
handover_event	Executes the handover: activates the target connection and resets its handover status
failed_handover_event	Used exclusively for logging a failed handover attempt by type; does not execute functional logic beyond history tracking
	RIC events
xApp_analyse_event	Triggers an xApp to execute its analysis, which determines potential control actions
xApp_control_event	Triggers an xApp to execute its control action based on a decision previously planned as a result of xApp_analyse_event
cd_analyse_event	Triggers the CD Agent (cd_ag) to identify conflicts among pending control decisions; if conflicts are found, it calls the CR Agent (cr_ag) to mitigate the conflicts according to currently configured CR method
postresolution_observation_event	Executes the post-resolution observation primarily used for RL in ML-based ConMit methods; calculates the final reward based on the network state after conflict mitigation and logs the training update
batch_training_event	Triggers the CR Agent (cr_ag) to execute a batch training step for the RL model and schedules the next batch_training_event
	PM & statistics events
process_kpi_event	Triggers a specific kpi object to process its measurements, calculate its current value, and schedule the next process_kpi_event for recurring measurements
save_stats_event	Triggers the RIC (ran_controller) to save the current global statistics of the network

Table 4.5: Events implemented in the simulator

Source: own elaboration

the simulator's configuration modules (specifically `config.py`) to ensure reproducibility of the experiments.

4.5 GUI operation mode

The basic mode of operation of the simulator is using a proprietary GUI. The main window of the GUI can be split into seven separate functional sections. These sections are marked in red color in Figure 4.1:

- GUI section 1 - drop-down menu with four categories of options:
  - Main - includes options to close the simulator and view basic information about the program,
  - Model - includes options to load and save ML models for ANN-based CR solutions,
  - Results - includes option to save results of an on-going simulation,
- GUI section 2 - visualization of the simulation area showing cells and users:
  - area of 4G LTE cells are colored blue,
  - area of 5G NR cells are colored red,

Parameter	Symbol	Value	Description
General environment			
Simulation area radius	R_{area}	450 m	The radius of the circular simulation field
Default simulation time	T_{sim}	1000 s	Duration of a standard simulation run
Mobility model			
Pedestrian speed	v_{ped}	6 m/s	Speed of users with pedestrian profile
Pedestrian ratio	r_{ped}	75%	Percentage of total users acting as pedestrians
Vehicle speed	v_{veh}	30 m/s	Speed of users with vehicular profile
Vehicle ratio	r_{veh}	25%	Percentage of total users acting as vehicles
LTE Macro configuration			
Middle frequency	$f_{\text{c,L-Mac}}$	800 MHz	Carrier frequency for LTE Macro cells
Bandwidth	$B_{\text{L-Mac}}$	10 MHz	Channel bandwidth
Tx Power	$P_{\text{tx,L-Mac}}$	32 dBm	Transmission power
Antenna height	$h_{\text{ant,L-Mac}}$	28 m	Height of the BS antenna
Antenna gain	$G_{\text{ant,L-Mac}}$	12.0 dBi	Gain of the BS antenna
Cells per BS	N_{cells}	3	Number of cells (sectors) per base station
Azimuth offset	ϕ_{offset}	180°	Starting azimuth orientation offset
Subcarrier spacing	Δf	15 kHz	Frequency separation between subcarriers
Subcarriers per RB	N_{sc}	12	Number of subcarriers per Resource Block
LTE Micro configuration			
Frequency	$f_{\text{c,L-Mic}}$	2100 MHz	Carrier frequency for LTE Micro cells
Bandwidth	$B_{\text{L-Mic}}$	20 MHz	Channel bandwidth
Tx power	$P_{\text{tx,L-Mic}}$	24 dBm	Transmission power
Antenna height	$h_{\text{ant,L-Mic}}$	12 m	Height of the BS antenna
Antenna gain	$G_{\text{ant,L-Mic}}$	4.0 dBi	Gain of the BS antenna
Cells per BS	N_{cells}	3	Number of cells (sectors) per base station
Azimuth offset	ϕ_{offset}	0°	Starting azimuth orientation offset
Subcarrier spacing	Δf	15 kHz	Frequency separation between subcarriers
Subcarriers per RB	N_{sc}	12	Number of subcarriers per Resource Block
NR Micro configuration			
Frequency	$f_{c,N-Mic}$	3500 MHz	Carrier frequency for NR Micro cells
Bandwidth	B_{N-Mic}	20 MHz	Channel bandwidth
Tx Power	$P_{tx,N-Mic}$	20 dBm	Transmission power
Antenna Height	$h_{ant,N-Mic}$	12 m	Height of the BS antenna
Antenna Gain	$G_{ant,N-Mic}$	4.0 dBi	Gain of the BS antenna
Cells per BS	N_{cells}	3	Number of cells (sectors) per base station
Azimuth offset	ϕ_{offset}	0°	Starting azimuth orientation offset
Numerology	μ	0	Subcarrier spacing configuration index
Subcarrier spacing	Δf	15 kHz	Calculated as $15 \cdot 2^{\mu}$ kHz
Subcarriers per RB	N_{sc}	12	Number of subcarriers per Resource Block
Handover & control parameters			
Default Time-to-Trigger	TTT	64 ms	Duration conditions must be met to trigger HO
Default handover hysteresis	$Hyst$	0 dB	Signal margin to prevent ping-pongs
Default Cell Individual Offset	CIO	0 dB	Default offset for cell selection

Table 4.6: Summary of simulation configuration parameters

Source: own elaboration

  - users are represented as darker dots, color each dot is associated with the user's traffic profile,
  - unique user is a dot with white middle,

- GUI section 3 - list of attributes of a cell chosen in list in GUI section 6,
- GUI section 4 - list of simulation parameters configurable via GUI, including parameters relevant to RL algorithms,
- GUI section 5 - list of values of received power from each cell for the unique user, used for troubleshooting at the early stage of implementation of the simulator,
- GUI section 6 - list of user-assigned names of cells deployed in the network,
- GUI section 7 - simulation controls:

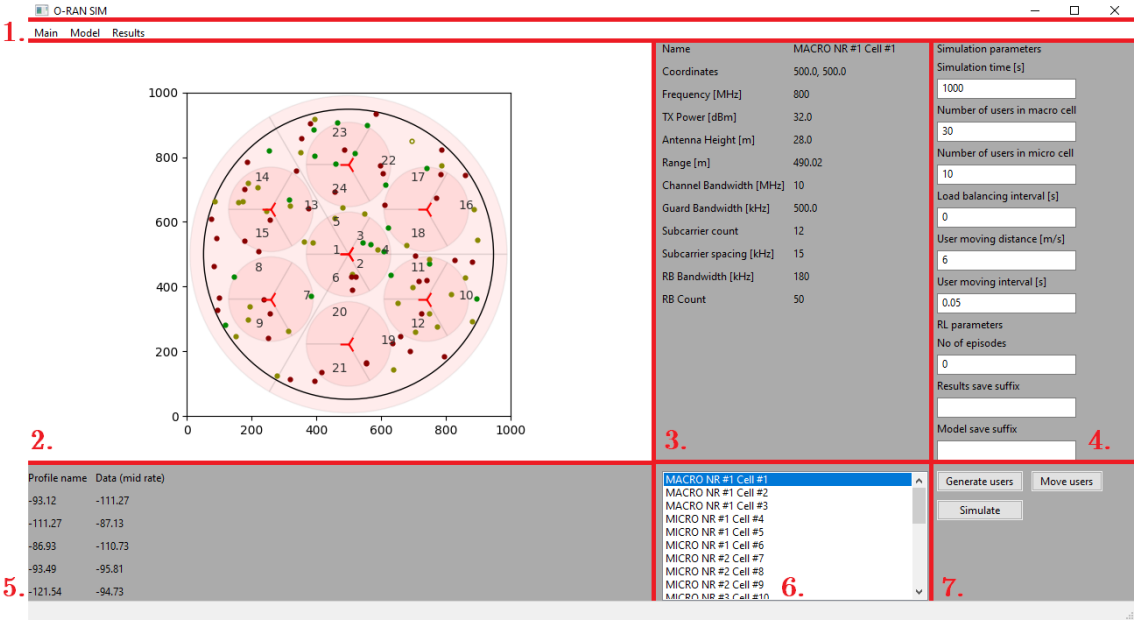

FIGURE 4.1: Main window of the simulator GUI

Source: own elaboration

- `Generate users` - generates users randomly placed in the simulation area according to parameters set in GUI section 4,
- `Move users` - moves all users in the simulation area according to parameters set in GUI section 4; used for testing at the early stage of implementation of the simulator,
- `Simulate` - starts the simulation according to parameters set in GUI section 4.

Configuration of the simulation using GUI starts by setting the proper parameters in GUI section 4. Then, users must be generated using the `Generate users` button. After pressing the button, any existing users are removed from the simulation area and new users are placed in random positions. All generated users become visible in the map in section 2 of the GUI. After the target placement of users is accomplished, the simulation process can be started by pressing the `Simulate` button. As the simulation progresses, simulation logs are printed in console. The logs are also printed into a temporary file, which is later renamed and moved to a dedicated directory once the simulation ends; see Section 4.7 for details of all output files. Conclusion of the simulation in GUI mode is signaled by a pop-up window including a set of information about the simulation process. This pop-up window is shown in Figure 4.2.

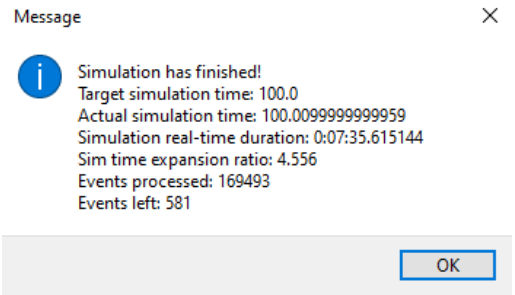

FIGURE 4.2: Pop-up window of the simulator GUI shown at the end of the simulation process

Source: own elaboration

4.6 CLI operation mode

The implemented simulation environment started as a purely GUI-based project. The main intent was to combine convenient configuration of the simulation with real-time visualizations of the simulation environment and key metrics. This approach was sufficient throughout the majority of the research conducted by the author. However, later stages of the project involving ML techniques required significantly more computation and an increased number of concurrently tested configurations. As a result, it became crucial to implement a mode of operation of the simulator that would enable automated parallel execution of preconfigured simulation runs, reducing the overall time required for running experiments. To that end, a Command Line Interface (CLI)-based mode of the simulator was developed.

The CLI mode allowed to assess the implemented solutions throughout the development process way more efficiently, mainly by reducing the efforts for manual supervision of the simulator. An additional benefit of implementing the CLI-based mode was that it became possible to utilize the processing center of the computing cluster deployed in Kąkolewo campus of Politechnika Poznańska, which is accessible remotely only through a console, with no possibility for convenient operation through GUIs.

There are two implemented ways of starting simulation runs using the CLI mode of the simulator:

- configuration via CLI - simulation run parameters are provided directly in CLI when starting the simulator; the scope of supported parameters is limited and only a single simulation run can be started per each CLI call. Example of CLI command for running the simulation using configuration directly in CLI with long and short options are shown in Listings 4.1 and 4.2, respectively.

Listing 4.1: Simulation run with configuration via CLI with long options

```
$ python module/main.py --name="simulation-run1" --time=1000 --ue-scenario=2 --model="ANN-
    model.pth"
```

Listing 4.2: Simulation run with configuration via CLI with short options

```
$ python module/main.py -n "simulation-run1" -t 1000 -u 2 -m "ANN-model.pth"
```

- configuration via configuration file - simulation run parameters are captured in a configuration file indicated in the CLI call of the simulator; this method enables scheduling multiple simulation runs running in series according to parameters included in the configuration file. If any parameter is not defined in a row of the configuration file, then a default value is used. Example of CLI command for running the simulation using configuration file is shown in Listing 4.3.

Listing 4.3: Simulation run with configuration via configuration file

```
$ python module/main.py -f run_parameters.csv
```

The second mode allows to run the simulations based on a Comma-Separated Values (CSV) file containing configuration of simulation runs. Each row in the file contains a set of parameters defining parameters that are variable across the considered scenarios. The parameters configurable via the CSV file are listed in Table 4.7. An example of this file is shown in Listing 4.4.

Table 4.7: Simulation configuration parameters

Short option	Long option / Config file parameter	Description	Example value
n	name	Identifier for the simulation run; used as suffix for result and model filenames	`-1-ANN5-network-S-10000s`
t	time	Duration of the simulation in seconds	`10000`
u	ue-scenario	Ordinal number of the UE deployment scenario	`1`
umac	ue-macro	Number of UEs per macro cell	`15`
umic	ue-micro	Number of UEs per micro cell	`6`
c	cr-method	Conflict resolution method (`DISABLED`, `PRIO_MLB_OMIT`, `PRIO_MRO_OMIT`, `PRIO_MLB_CD`, `PRIO_MRO_CD`, `ANN`)	`ANN`
a	ann-version	Version of ANN to use when the conflict resolution method is ANN	`5`
e	eps	Initial exploration factor (ε) for ANN training	`1.0`
d	decline-eps	Epsilon decay rate for ANN training	`0.9999`
g	gamma	Discount factor γ for reinforcement learning	`0.1`
l	lr	Learning rate α for ANN training	`1.0e-5`
m	model	Filename of the model to load for the simulation run; special value `latest` indicates usage of last saved model	`latest`
f	file	CSV file containing configuration parameters for batch runs; parameter is not supported in configuration parameter file	`config.csv`
–	neuron-count-scale	Scale multiplier for the number of neurons in each ANN layer	`1.0`
–	neuron-count-base	Base number of neurons per layer before scaling	`128`
–	tau	Soft update coefficient τ for the target network	`0.1`
–	weight-decay	L2 regularization coefficient for weight decay	`1e-4`
–	dropout-factor	Dropout probability for ANN layers	`0.1`
–	huber-loss-delta	δ parameter for the Huber loss function	`1.0`
–	cm-reward-mode	Reward type for CM algorithm (`NETWORK`, `CONTROL_TARGET`, `CONTROL_TARGET_COUNTS`)	`NETWORK`
–	cm-reward-normalization	Normalization method for CM reward (`EPOCH_MIN_MAX`, `EXP_MOVING_AVERAGE`)	`EPOCH_MIN_MAX`
–	cm-reward-exp-moving-average-alpha	Exponential moving average factor for reward normalization in `EXP_MOVING_AVERAGE` mode	`0.1`
–	cm-ann-rl-method	Reinforcement learning algorithm for CM ANN (`Q-LEARNING`, `SARSA`); not applicable in ANN version 5, where PPO-Clip algorithm is used	`Q-LEARNING`
–	lr-mode	Learning rate scheduling mode (`STATIC`, `ADAPTIVE`)	`STATIC`
–	max-grad-norm-clip	Maximum gradient norm for clipping during ANN training	`0.5`
–	evaluation-mode	Enables evaluation-only mode (no learning) if set to `True`	`True`
–	rl-rollout-length	Number of simulated seconds per RL rollout for batch training	`400.0`

Listing 4.4: Example CSV file containing parameters for configuration of simulation runs

```
ue-scenario,name,time,ue-macro,ue-micro,cr-method,ann-version,eps,decline-eps,gamma,lr,
    model,neuron-count-scale,neuron-count-base,tau,weight-decay,dropout-factor,huber-loss
    -delta,cm-reward-mode,cm-reward-normalization,cm-ann-rl-method,cm-reward-exp-moving-
    average-alpha,lr-mode,max-grad-norm-clip
1,-1-noCM-S-1000s,1000,15,6,DISABLED,,,,,,,,,,,,,,,,,
1,-1-prioMLB-CD-S-1000s,1000,15,6,PRIO_MLB_CD,,,,,,,,,,,,,,,,,
1,-1-prioMRO-CD-S-1000s,1000,15,6,PRIO_MRO_CD,,,,,,,,,,,,,,,,,
1,-1-ANN5-network-S-10000s,10000,15,6,ANN,5,1.00,0.9999,0.1,1.0e-5,,1.0,128,0.1,1e-4,0.1,
    1.0,NETWORK,EPOCH_MIN_MAX,Q-LEARNING,,STATIC,0.5,False,400.0
1,-1-ANN5-network-S-1000s-eval-after-10k,1000,15,6,ANN,5,1.00,0.9999,0.1,1.0e-5,latest,1.0
    ,128,0.1,1e-4,0.1,1.0,NETWORK,EPOCH_MIN_MAX,Q-LEARNING,,STATIC,0.5,True,400.0
```

4.7 Outputs

Regardless of the used operation mode (GUI or CLI), the simulator produces a set of files. The simulation log file containing messages logged in console during the simulation process is used for troubleshooting, while the CSV files capturing network-level metrics throughout the simulation

process are used for evaluation of tested CR solutions. Name of the simulation log file comprises the timestamp of when simulation ended and the value of `Results save prefix` configured in GUI section 4. Same naming convention is used for files containing key metric results of the simulation.

The simulator also supports some ConMit modes, which implement ML techniques for the CR processing. When using these modes, the ML model trained throughout the simulation process is automatically saved at the end of each simulation run. Name of the model file in *.pth* format is constructed using the timestamp captured when simulation concluded and the value of `Model save suffix` configured in GUI section 4.

The produced files are listed and described in Table 4.8.

Table 4.8: Simulation output files and their contents

Filename pattern	Description	Contents/columns
simulation_*<date>*_*<suffix>*.log	Main simulation log capturing console output in both GUI and CLI simulation modes, contains initialization messages, runtime information, and periodic statistics-saving entries; useful for debugging and monitoring overall progress	timestamps, log level, event messages
satis_*<date>*_*<suffix>*.csv	Average user bitrate satisfaction percentage across the network	time, satisfaction
avail_*<date>*_*<suffix>*.csv	Time series of average cell availability across the network; provides an overview of network accessibility and stability over time	time, availability
lb_*<date>*_*<suffix>*.csv	Average Jain's fairness index for load balancing between cells; reflects how evenly traffic load is distributed across cells	time, lb ratio
tpt_*<date>*_*<suffix>*.csv	Total throughput of all cells in the network at each time step	time, throughput
ho_*<date>*_*<suffix>*.csv	All handover events occurring during the simulation, including timestamps, source/target cells, user identifiers, and radio conditions	time, previous cell, current cell, user, conn_sinr, x pos, y pos
pp_*<date>*_*<suffix>*.csv	Ping-pong handover events where a user returns shortly to a previous cell; useful for assessing handover instability	time, current cell, user, conn_sinr, x pos, y pos, ho pp time
cell-*<BS ID>*_*<cell ID>*_*<date>*_*<suffix>*.csv	Per-cell log of handover configuration parameters; one file is generated for each simulated cell	time, bs, cell, availability, cio, hyst, ttt
cb_*<date>*_*<suffix>*.csv	Call blockage events indicating failed connection attempts due to lack of resources or coverage; includes user positions for spatial analysis	time, user, x pos, y pos
rlf_*<date>*_*<suffix>*.csv	Radio link failure log detailing user disconnections due to poor radio conditions; each entry contains user ID, serving cell, and position at failure time	time, current cell, user, conn_sinr, x pos, y pos
*<date>*_*<model suffix>*.pth	File containing weights and biases of a trained ML model saved at the end of each simulation run using ML-based CR methods	parameters (including weights and biases) of the PyTorch model

Chapter 5

Proposed approach for O-RAN conflict mitigation

5.1 Conflict mitigation activities categorization

To simplify communication between parties working on issues regarding conflict mitigation, it is important to introduce a categorization for the various ConMit activities in O-RAN. The following categorization was originally proposed by the author in 2023 [1]:

1. Preventive conflict mitigation – activities aimed at avoiding conflicts before they occur; activities that are part of the preventive conflict mitigation control loop aim to avoid RAN control conflicts before they occur in a live network; this category includes, but is not limited to, pre-deployment activities and dedicated RAN control modes that limit the possibility of conflicts; further described in Section 5.3,

2. Conflict detection and resolution (CD&R) – activities aimed at detecting conflicts, if they occur despite preventive measures, and resolving them in an optimal way; further described in Section 5.4,

3. Supervision and adaptation (S&A) – activities aimed at monitoring, maintaining, and reconfiguring conflict mitigation components to ensure proper network operation and alignment with policies applied by the MNO maintaining the network; further described in Section 5.5.

It is important to highlight that this analysis was a novel contribution to the field, published significantly earlier than the subsequent O-RAN Alliance technical reports regarding conflict mitigation [17]. The proposed extensions to O-RAN architecture and control loops related to the proposed categories of ConMit activities listed above are described in the following sections.

Note that parts of this categorization is semantically in line with categorization of ConMit procedures later provided in O-RAN Alliance's technical report about ConMit [17], which was mentioned in Section 3.4.1. However, it needs to be highlighted that:

- as mentioned above, the article describing the above categorization [1] was published 14 months earlier than the O-RAN Alliance technical report [17]; the article [1] was the first published work on the topic of challenges in O-RAN ConMit,

- the report lists "Conflict Detection" and "Conflict Resolution" as separate procedures, while in the article they are contained within a single category of ConMit activities "conflict detection and resolution",

- the report defines "Conflict Avoidance" procedure as providing guidance and additional information to xApps to avoid conflicts, while the article defines "preventive conflict mitigation" as a broader category of activities for conflict avoidance.

5.2 Extensions to O-RAN architecture

The author's proposition for ConMit operation includes several extensions to the O-RAN architecture. To ensure efficient and reliable operation of the network, several new logical entities are introduced within the Near-RT RIC's ConMit component: CD Agent, CR Agent, Performance Monitoring (PMon), and ConMit Supervisor. The first three components are primarily responsible for executing the CD&R activities for ConMit, which are detailed in Section 5.4. The ConMit Supervisor, in contrast, is the main agent for the S&A activities, as discussed in Section 5.5. The O-RAN architecture with the proposed extensions is illustrated in Figure 5.1. In this diagram, the specific ConMit components proposed by the author are highlighted in light blue.

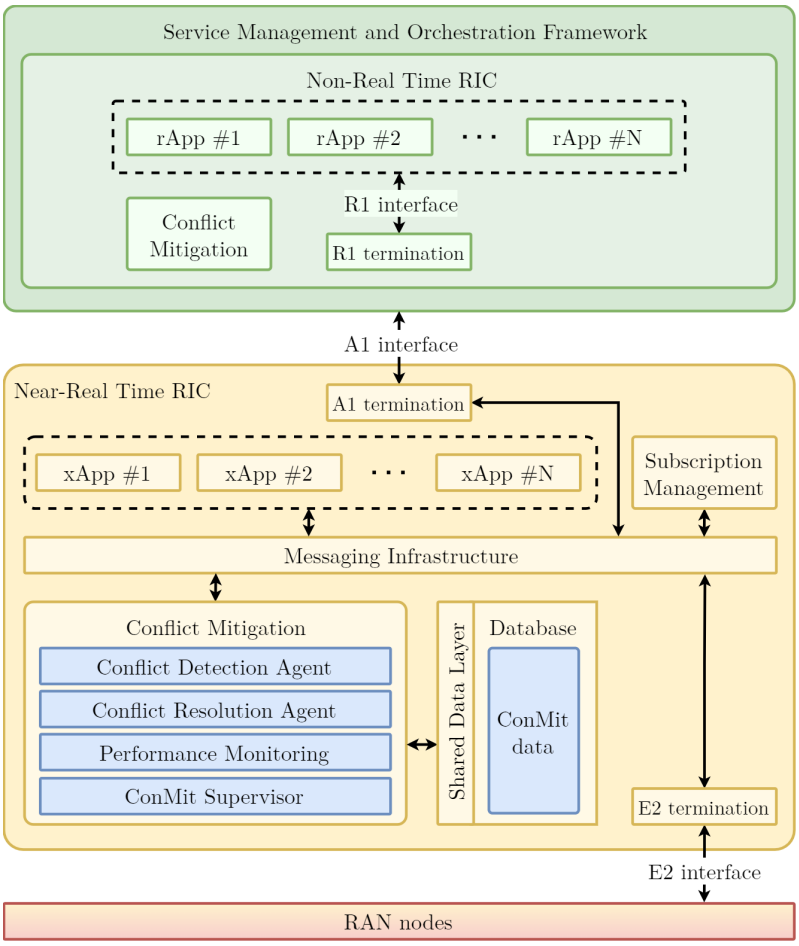

Figure 5.1: O-RAN architecture with the proposed ConMit components

Source: own elaboration

Moreover, the author proposes an additional scope of ConMit-related information to be exchanged between the network components, utilizing the open interfaces already existing in the O-RAN architecture.

The proposed extensions are described in the following sections.

5.2.1 Conflict Detection Agent

The CD Agent is a logical component responsible for detecting conflicts between various control agents deployed in the network. It may employ various techniques such as anomaly detection, correlation analysis, and pattern recognition to identify conflicts, collect relevant data, and report these findings to other ConMit components. As such, the CD Agent's actions serve as the first step in mitigating conflicts that occur in runtime of the network. For that reason, the CD Agent must be able to detect all three types of control conflicts expected in O-RAN: direct, indirect, and implicit. The specific methods for detection of conflicts proposed by the author are described in Section 5.4.3.

Other than the simple conflict detection, the CD Agent may additionally compute an evaluation for each conflict. It is expressed as a metric or a set of metrics describing the conflict's impact, either on the network as a whole (conflict severity) or on a specific relevant parameter or metric. This evaluation can then be used by other ConMit components, such as the CR Agent and ConMit Supervisor, to improve the overall ConMit process and provide the operator with deeper insights into detected conflicts.

For example, the CD Agent may calculate a severity score for a detected conflict based on the criticality of the target cell (e.g., a cell serving emergency users). This evaluation is passed to the CR Agent. If the severity score exceeds a high-priority threshold, the CR Agent may opt for an immediate, failsafe resolution (e.g., rejecting all changes) rather than attempting a complex negotiation, thereby preserving network stability in critical scenarios.

A graphical representation of the CD Agent showing its functions together with its inputs and outputs is shown in Figure 5.2.

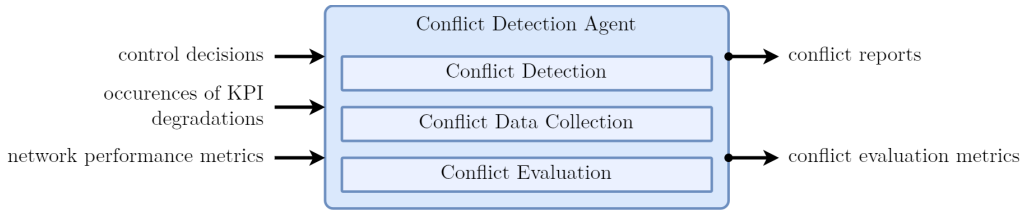

FIGURE 5.2: Functions, inputs, and outputs of the proposed Conflict Detection Agent component

Source: own elaboration

5.2.2 Conflict Resolution Agent

The CR Agent is responsible for resolving conflicts identified by the CD Agent. It is designed to handle all control conflicts that arise during network operation, irrespective of the conflict type, the conflicting applications, or the specific conflicting control decisions.

The CR Agent employs dedicated logic to determine the optimal course of action. While achieving a truly optimal solution should always be the target of a CR method, it is expected to be unfeasible in the vast majority of cases. As such, the CR Agent must find the best possible resolution within the available resources and computation time. Ideally, when computing the target CR actions, the implemented solution should consider the conflicting control actions, current network conditions (including policies provided by the Non-RT RIC), past ConMit experiences, and additional information from other ConMit components. The CR Agent is also responsible for executing the configuration changes and control decision modifications it devised to resolve the conflict. The specific methods for resolution of conflicts proposed by the author are described in Sections 5.4.5 and 6.2.

Consider a scenario where xApp #1 (MRO) and xApp #2 (ES) trigger a direct conflict on antenna tilt. The CR Agent receives the relevant conflict report and retrieves info about its current policy. If the this policy dictates that "user experience is more important than power saving", the CR Agent applies logic to accept the decision from xApp #1 and reject xApp #2.

A graphical representation of the CR Agent showing its functions together with its inputs and outputs is shown in Figure 5.3.

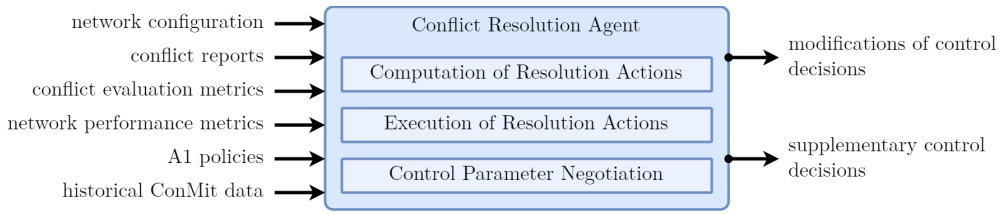

FIGURE 5.3: Functions, inputs, and outputs of the proposed Conflict Resolution Agent component

Source: own elaboration

5.2.3 Performance Monitoring

The PMon component is responsible for monitoring RAN performance metrics in network runtime. Its primary objective is to continuously analyze these metrics to identify trends, anomalies, and deviations from the expected baselines. Based on these observations, the PMon component reports any findings for consideration by other ConMit components. Additionally, the insights gathered by the PMon component can be utilized by the RIC for purposes beyond ConMit, such as evaluating the effectiveness of specific deployed applications.

The responsibility of PMon is limited to reporting anomalies in observed KPIs; it does not determine which control decisions correlate with the detected degradations of the performance metrics. However, this KPI analysis function of PMon can be enhanced by considering all control decisions issued by the applications. This input can be used, for example, to decide which KPIs to monitor and how often to poll their values.

The CD Agent, for instance, may use the results of PMon's analysis for the purpose of detecting implicit conflicts. As a primary example of this interaction, PMon would detect and log occurrences of notable KPI deterioration. The CD Agent, receiving this report and also being aware of the active control decisions at that time, can then perform its own analysis. If it identifies a pattern of such degradations repeatedly occurring after similar sets of control decisions, the CD Agent can determine that the observed behavior is a result of an implicit conflict. In practice, PMon can maintain a rolling window of KPI values (e.g., handover success rate observed over the last 10 seconds). It compares the current moving average against a pre-configured degradation threshold (e.g., dropping below 95%). Upon a breach, PMon generates a notification containing the information identifying the specific control target and KPI, together with metrics describing the observed anomaly. Such notification is immediately dispatched to the CD Agent for correlation analysis. This mechanism is further described in Section 5.4.3.

A graphical representation of the PMon component showing its functions together with its inputs and outputs is shown in Figure 5.4.

5.2.4 Conflict Mitigation Supervisor

The ConMit Supervisor is a logical component that monitors ConMit activities within the RIC and tracks their impacts on network metrics. Its role is to report on the effectiveness of deployed

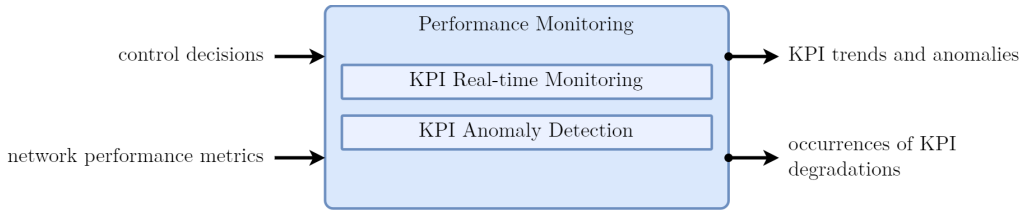

FIGURE 5.4: Functions, inputs, and outputs of the proposed Performance Monitoring component

Source: own elaboration

ConMit measures and implement necessary configuration updates to adapt these components to the current network conditions. For example, it may provide configuration guidance to the CD Agent regarding the target sensitivity for implicit conflict detection. The ConMit Supervisor also aids in enforcing A1 policies provided by the Non-RT RIC. It does this by altering the parameters of ConMit components to align their behavior with the current policy. For instance, adjusting the CR Agent's application priority to align its behavior with the desired performance targets from the policy.

A graphical representation of the ConMit Supervisor showing its functions together with its inputs and outputs is shown in Figure 5.5.

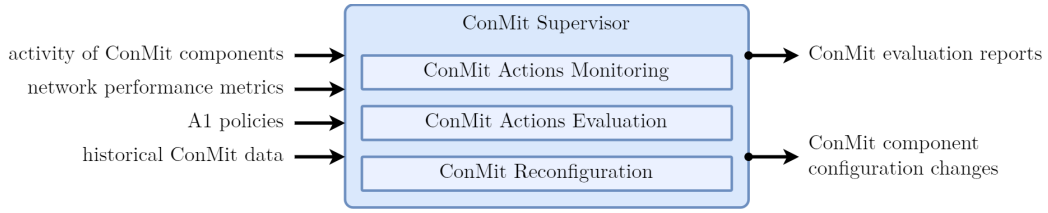

FIGURE 5.5: Functions, inputs, and outputs of the proposed Conflict Mitigation Supervisor component

Source: own elaboration

5.2.5 Exchange of control and ConMit information

Other than introduction of the new components, the author's proposal for ConMit includes an expanded scope of information exchanged between the RICs. The motivation for this extension is to mitigate the inter-Near-RT-RIC conflicts by making neighboring Near-RT RICs aware of each other's control decisions. Specifically, this approach ensures that each Near-RT RIC has knowledge about all control decisions that may influence network elements along the shared boundaries of their respective managed areas. To accomplish this, each Near-RT RIC may share information about its active control decisions with the Non-RT RIC via the A1 interface, as part of the A1 enrichment information. The Non-RT RIC can then distribute the information to all other relevant Near-RT RICs. Once the Near-RT RICs are aware of decisions for control targets managed by their neighbors, these external decisions can be considered as part of their local ConMit processing.

Consider two neighboring cells, A and B, managed by distinct Near-RT RICs (#1 and #2). In a handover control scenario, if RIC #1 modifies parameters for Cell A, it directly affects the handover boundary with Cell B. Without coordination, this leads to unmitigated configuration conflicts, as both RICs impact each other's cell's handover boundary. Similarly, in an interference scenario, if an xApp in RIC #1 increases Cell A's transmission power, it may degrade SINR in Cell B. Typically, RIC #2 would perceive this as random external noise. However, with the proposed extension, RIC #1 shares its decision via the A1 interface. This allows RIC #2 to correlate the performance drop in Cell B with the specific action taken by RIC #1, effectively detecting the

inter-RIC conflict. In both instances, shared visibility enables the ConMit component to resolve inter-RIC conflicts.

Separately from the communication for mitigating inter-Near-RT-RIC conflicts, the operation of the proposed ConMit extensions relies on ConMit-related data exchanged between the new components. Since there are no dedicated interfaces for this purpose, these components may communicate via the Near-RT RIC's Database, which is accessible through the SDL as specified in the O-RAN standard. An alternative approach, suitable for when ConMit is implemented as a unified component, is to use internal interfaces, avoiding indirect communication via the database.

Regardless of the implemented approach, the proposed ConMit components must store ConMit-related information in the database for data persistence. The scope of ConMit data stored in this database includes, but is not limited to, information about detected conflicts, resolution actions, evaluation metrics, and observed KPI anomalies. Storing this ConMit-related information serves a dual purpose: it enables the core operation of the proposed extensions and ensures that a complete log of all significant ConMit events is maintained, which can be crucial for troubleshooting network operation in case of failures.

5.3 Preventive conflict mitigation

The majority of the contributions presented in literature focus mostly on the reactive mitigation of control conflicts during network operation. However, building a truly robust solution for ConMit requires the inclusion of preventive activities that aim to detect and address conflicts before a set of applications is deployed in the network. Furthermore, while an operator may accept a certain level of control conflict during network operation in normal conditions, it is important to provide means for the operator to temporarily enforce stricter mitigation policies in cases where network conditions become challenging and avoiding conflicts is crucial. This section proposes several solutions for preventive ConMit activities, covering both pre-deployment conflict assessment and dedicated runtime operational modes designed for conflict avoidance during critical network conditions.

As a key pre-deployment activity, the author proposes assessing impact of an application (rApp, xApp, or dApp), that is being considered for deployment, in a lab or simulated environment. Ideally, the simulated network should align with the concept of digital twin, meaning that the digital representation of the network should mirror the original network's behavior. This approach uses CD&R capabilities of the proposed conflict mitigation toolset (see Section 5.4 for the details) to detect any conflicts happening in the network upon adding a new application to the environment. Assessing the impact of each application considered for deployment provides MNOs with insight into the pros and cons of deploying a specific application. With this knowledge, the MNO can decide to either not deploy the application, reconfigure the existing application setup in the network to remove any potentially conflicting applications, or deploy the new application regardless of any conflicts, accepting any observed drawbacks.

To avoid conflicts in a live network with multiple applications deployed, two dedicated modes of operation for the ConMit component of RICs are proposed by the author [1]. The first mode is the *conflict avoidance mode*, in which each control message provided by an application in a RIC is first provided to the ConMit component. Then, the logic of the ConMit component (as part of its CR capabilities) decides whether the control decision should become active in the network, taking into account the network performance, prior control decisions, and the current policies set by the MNO. The conflict avoidance mode aims to limit the possibility of conflict happening in the network by ensuring that the ConMit component can assess and block any control decision considered as potentially conflicting before it is applied in the live network.

The second dedicated mode of operation of the ConMit components is the *critical pipeline mode*. Upon receiving a critical KPI alert from the Performance Monitoring component, the ConMit Supervisor can enforce a specific order in which xApps/rApps provide their control decisions to mitigate the issue that triggered the KPI alert. This mode of ConMit operation is aimed at enabling efficient network reconfiguration upon critical deterioration of the network performance. For example, the applications implementing self-healing capabilities of the network may be prioritized in the critical pipeline mode.

It is important to highlight that dApps are ignored from consideration in the critical pipeline mode, as they operate in real time control loop and, as such, cannot be reliably controlled by Non-RT and Near-RT RICs. As such, dApps can either be disabled or enabled during the critical pipeline mode, depending on the MNO-specific configuration.

The preventive ConMit activities may also include exchange of additional information between network components (RICs, xApps) to aid xApps in making non-conflicting control decisions. At the moment, such interface is not envisioned as part of O-RAN specifications.

A diagram illustrating the preventive conflict mitigation control loop is shown in Figure 5.6.

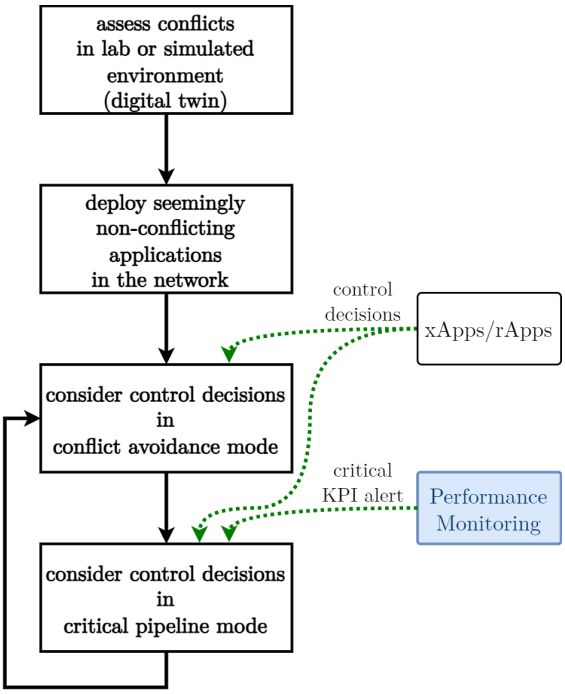

FIGURE 5.6: Preventive conflict mitigation control loop

Source: own elaboration

5.4 Conflict detection and resolution

Control conflicts occurring in runtime can degrade or even nullify the positive impact of optimization [1]. For example, if one xApp increases tilt to improve coverage and another decreases it to reduce interference, the physical antenna parameter may oscillate or settle at a value that satisfies neither objective, effectively canceling out the optimization gains intended by both applications. Therefore, the conflicts should be avoided where possible with proper network setup [15] via preventive conflict mitigation activities, as showcased in the previous section. However, the complex dependencies between RAN control activities and evolution of the network throughout its lifetime making it not viable to completely mitigate control conflicts with a static configuration alone. Thus,

conflicts that appear during network operation must be detected and resolved dynamically. For that purpose, the O-RAN specifications envision ConMit components in the Near-RT RIC and in the Non-RT RIC [44, 45]. Using the ConMit extensions proposed by the author, these runtime conflicts can be mitigated effectively.

Reliable detection of conflicts is a prerequisite for effective mitigation of conflicts occurring during network operation. As described in Section 5.2.1, the author proposes the CD Agent as the primary component for this task, responsible for detecting all control conflicts within the O-RAN deployment. Upon detecting a conflict, the CD Agent must collect all data required for its assessment and resolution. Furthermore, the agent can perform additional processing of the relevant information to evaluate the conflict's severity. This evaluation provides the MNO with quantified insight into the negative impact of conflicts and supplies data for fine-tuning of the ConMit mechanisms (as described further in Section 5.5).

Once a conflict is detected, it must be resolved to minimize its negative impact on network operation. The authors envisions the CR Agent as the component responsible for conflict resolution. It must adhere to the Optimization principle (described in Section 3.3): the implemented CR method must not worsen (and ideally should improve) network performance and reliability compared to a scenario without conflict resolution. The CR Agent can take a range of actions towards the conflicting control decisions to resolve the conflict:

- application - control decision is applied without any alteration,
- rejection - control decision is rejected completely and is not applied in the network,
- modification - control decision is applied with a modification to its parameters.

Beyond the direct manipulation of conflicting control messages, the CR Agent can take broader ConMit actions that, for example, impact the lifecycle of applications or the state of network elements:

- temporarily preventing applications from providing control decisions for the control target - conflicting application is put on cooldown, during which it cannot provide new control decisions towards the control target,
- disabling applications - conflicting application is completely disabled,
- disabling of network elements - a network element related to the conflict (e.g., O-RU) is disabled,
- negotiate a parameter value - CR Agent communicates with conflicting applications to compute an optimal value of a parameter, that should ideally meet optimization goals of all parties,
- creating control decisions - CR Agent generates new control decisions, which are supplementary to the conflicting decisions.

The actions listed above are only examples observed in the literature and are not exhaustive. There are no explicit restrictions identified in O-RAN specifications regarding the scope of CR actions available to RICs' ConMit components. A conflict resolution computed by the CR Agent may comprise a set of CR actions affecting control decisions, control targets, and applications relevant to the detected conflict. However, these actions can also extend beyond the directly affected elements, influencing other parts of the network related to the conflicting control decisions. For

example, the ConMit component may decide to adjust the CIO of a neighboring cell to offload traffic from a conflict-affected cell, which in turn mitigates negative impacts of the conflict.

Both CD and CR methods can use AI/ML techniques. The motivation is to use their capability to adapt to observed conditions to derive such ConMit policies that would optimally address the conflict case. Specifically, one of the crucial use cases for AI/ML is the detection of implicit conflicts. These conflicts cannot be easily detected pre-action, as the relations between control parameters and affected KPIs and other parameters are not trivial. Employing ML measures enables efficient computation of large datasets related to network metrics following control decisions, allowing detection of consistent negative influences related to specific control decisions. Such approach for ML-driven CD using GNNs is already described in the literature [142, 146], along with evaluation of the method showing its promising results.

To contextualize the author's proposed AI-based solution, it is relevant to briefly revisit key approaches described in Section 3.4.3. The utilization of AI/ML in CR methods is expected to be more diverse than in CD, as the range of possible CR actions is considerably broader and the underlying decision process inherently more complex. With AI/ML, the CR solutions can move beyond static rule-based approaches and instead determine (near-)optimal resolution actions based on a large set of observed factors. Several such methods have already been proposed in the literature. For instance, Cinemre and Mahmoodi utilize ML to decide which applications to deploy at a given time [24]; Zafar et al. use DDQN to dynamically disable or enable O-RUs [143]; and Erdol et al. employ policy distillation to merge a set of conflicting applications into a single optimized one with the same action set [20]. Other promising applications of ML-driven CR include generating supplementary control decisions to support conflict resolution, placing applications on temporal cooldown, or modifying conflicting control decisions. The latter two actions form the basis of the ML-based CR approach proposed by the author [6], which is described in detail in Chapter 6.

5.4.1 CD&R control loop

The proposed control loop for CD&R utilizes the proposed CD Agent and CR Agent components to detect and optimally resolve all types of conflict types occurring in the network. The CD Agent monitors the control decisions of applications for potential conflicts. To perform this monitoring, the Messaging Infrastructure of the RIC must be configured to intercept all E2 Control messages and forward them to the ConMit component before they are dispatched to the E2 Termination. The CD Agent extracts the target E2 Node ID and the list of modified parameters, cross-referencing this data against the database of currently active control actions. Upon conflict detection, the CD Agent reports relevant conflict reports to the CR Agent

The CR Agent then analyzes these reports and decides on the best course of action to resolve the conflicts. Both agents use the RIC's database to log information about their ConMit activities. The control loop is closed by the RAN nodes, which report performance metrics to the applications (for further optimizations) and to the proposed PMon component (for KPI anomaly detection). This process enables the CD and CR Agents to continuously monitor what is happening in the network and react to mitigate any control conflicts. The control loop for CD&R activities is shown in Figure 5.7.

5.4.2 Conflict Mitigation Framework

O-RAN specifications define two dedicated procedures for ConMit in the Near-RT RIC: E2 Conflict Mitigation Guidance and E2 Conflict Mitigation Assistance [17, 45]. As described in

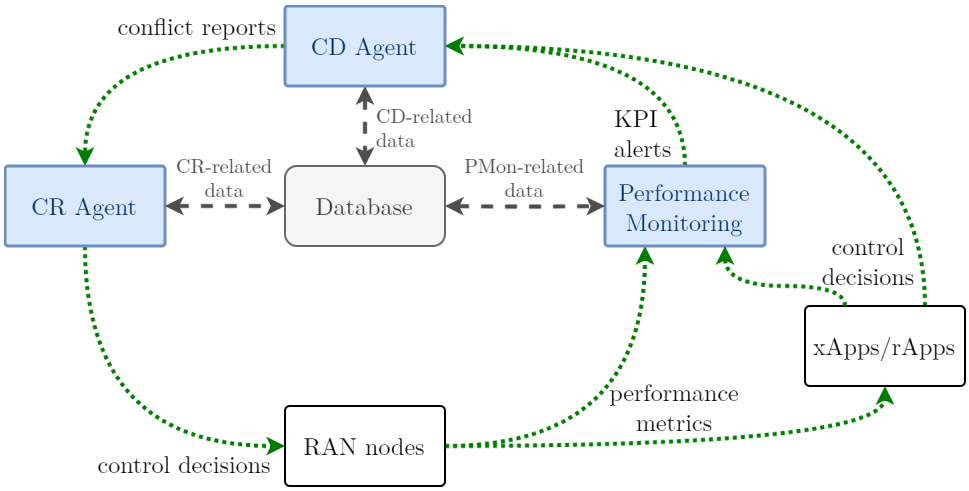

FIGURE 5.7: Conflict detection and resolution control loop

Source: own elaboration

Section 3.4.1, these standard O-RAN mechanisms require complex, dedicated ConMit logic in both the RIC and the applications to handle diverse conflict scenarios. Moreover, the required back-and-forth communication between these components can introduce significant delays between conflict detection and resolution. Consequently, the author considers these procedures insufficiently robust for effective conflict mitigation. Furthermore, the O-RAN specifications do not provide any details about logic for detection of three standard types of conflicts, leaving both the CD and CR completely up to the operator.

The author proposes the Conflict Mitigation Framework, or CMF, as an alternative approach for implementing the CD&R control loop, replacing the standard O-RAN-specified ConMit mechanisms. The initial concept of this framework was first presented in a conference paper in Polish language [7], and later published in an article in English [2], with extended descriptions and simulation-based evaluation results.

The CMF comprises a set of procedures dedicated to detection and resolution of control conflicts, using the proposed ConMit components described in Section 5.2. Specifically, definitions of procedures in CMF assume the existence of CD Agent, CR Agent, and PMon, implemented in accordance with the descriptions provided previously.

The procedures defined as part of CMF imply the methods for detection of all three standard types of conflicts. These CD procedures are described in detail in Section 5.4.3. In contrast, the CMF operation is agnostic to the applied CR method and, as such, does not restrict the resolution logic in any way. Basic rule-based CR algorithms proposed by the author are presented in Section 5.4.5, while an advanced ANN-based method is showcased in Section 6.2.

A key advantage of the proposed framework is that it does not impose any new requirements on the applications themselves. It only requires that all E2 control messages are forwarded by the RIC's Messaging Infrastructure to the ConMit component, rather than being sent directly to the target E2 RAN nodes. As a result, in contrast to the E2 Conflict Mitigation Guidance and Assistance procedures defined in the O-RAN Alliance specifications, the CMF acts as a gateway for all E2 control decisions. This design eliminates the need for any ConMit-specific logic to be implemented in the deployed applications. A comparison of the ConMit operation in the standard O-RAN procedures and the proposed CMF solution is shown in Figure 5.8.

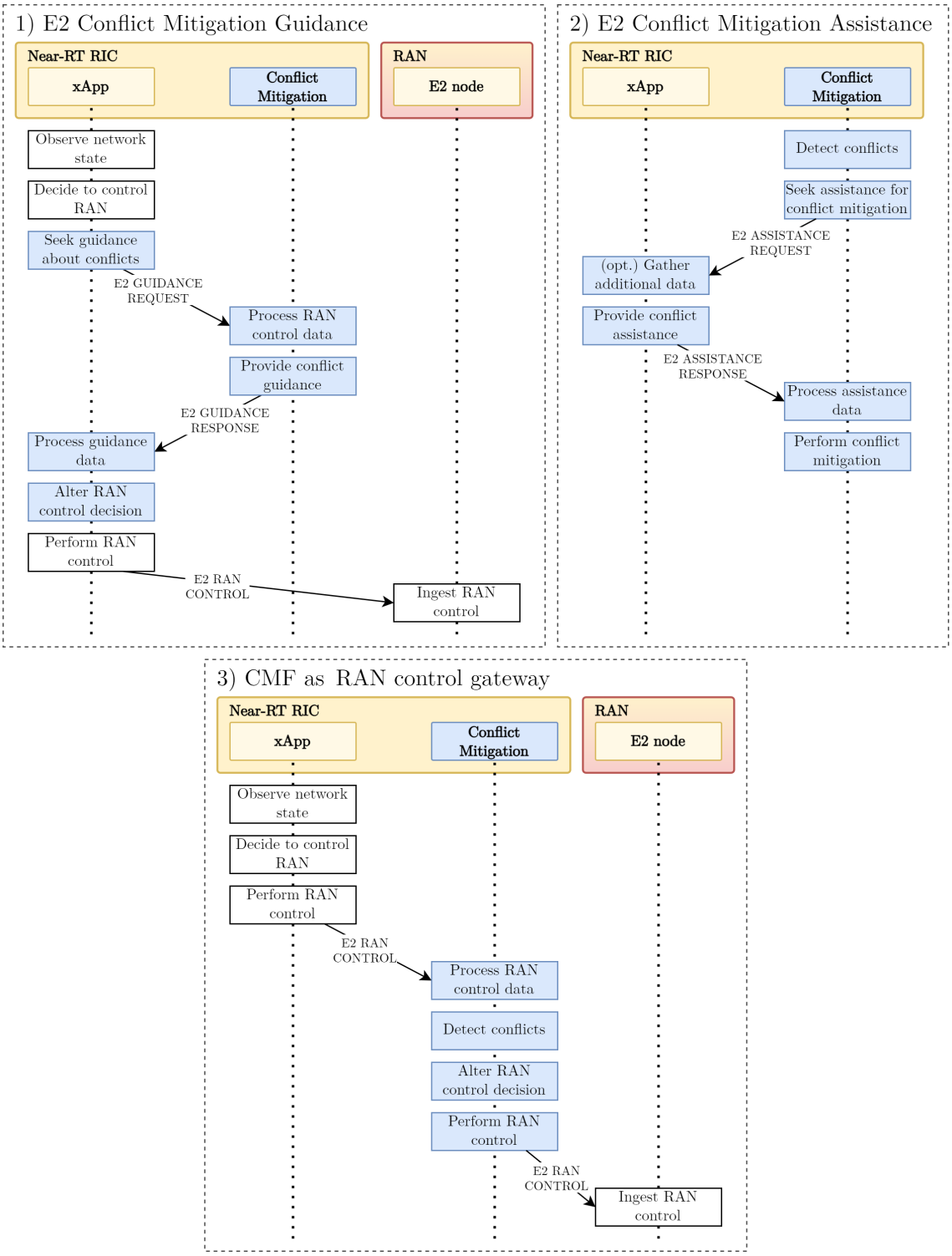

FIGURE 5.8: Comparison of ConMit approaches specified in O-RAN Alliance specifications with the Conflict Mitigation Framework approach

Source: own elaboration based on [17, 45] [2]

5.4.3 Proposed conflict detection procedures

The CD procedures described below are designed and proposed by the author to cover the three O-RAN conflict types: Direct Conflict Detection (DCD), Indirect Conflict Detection (ICD), and Implicit Conflict Detection (ImCD). During their execution, the CMF's components for the CD&R control loop (i.e., CD and CR Agent, PMon) utilize the RIC's database to store and

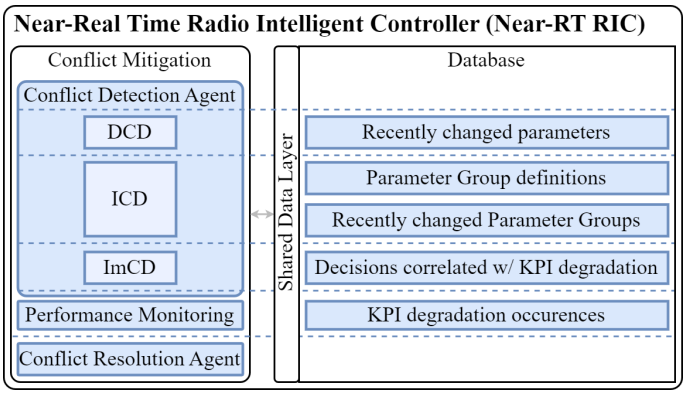

FIGURE 5.9: Relations between components involved in CMF conflict detection procedures and the data stored in the Near-RT RIC's database

Source: own elaboration

access ConMit-related information, as specified in Section 5.2.5. This data is required for the CD procedures and is subsequently used by the implemented CR method. The scope of the data utilized and stored by each specific CD procedure is described in the following paragraphs.

A key benefit of the proposed information exchange extension described in Section 5.2.5 is that Near-RT RICs share knowledge about control decisions made within each RIC. This way, the proposed CD procedures can also consider control decisions from neighboring RICs to detect any inter-Near-RT-RIC conflicts.

The CD procedures in the CMF store ConMit-related data in the RIC's database. Figure 5.9 illustrates the relations between the specific data entries and their main associated component or procedure within the CMF, though other components may also access this data. In the figure, the CD procedures are shown as part of the CD Agent, as it is the main component executing the detection process.

Direct Conflict Detection

The proposed DCD procedure is designed to detect direct conflicts. This procedure operates as a pre-action check, meaning it detects the conflict before a control decisions that would cause it is applied to the network. For this purpose, the CD Agent tracks all control messages in the RIC's database (see "Recently changed parameters" in Figure 5.9). Each control decision is logged along with its timestamp, source application, control target, modified parameters, and control time span (i.e., the control duration expected by the application). Once it is stored in the database, the control decision is analyzed against the set of currently active decisions.

The conflict detection logic of DCD is as follows: each new control message in the Near-RT RIC is logged into the Database, then it is compared to all currently active control decisions. If the new control decisions shares the same control target and at least one of the modified parameters with any of the active decisions, data about the conflicting decisions is provided to the CR Agent. The DCD procedure is illustrated in Figure 5.10, which shows the message exchange flow between the CMF components. An example information exchange for the proposed DCD procedure is presented in the following paragraphs.

First, an xApp issues a control decision intended for a network element. Instead of going directly to the target, this E2 control message is intercepted by the CMF through routing configuration of the Messaging Infrastructure. As presented in Listing 5.1, the message contains the source application (xApp1), the command (Modify), and the target parameter (TxPower) with its new value (32).

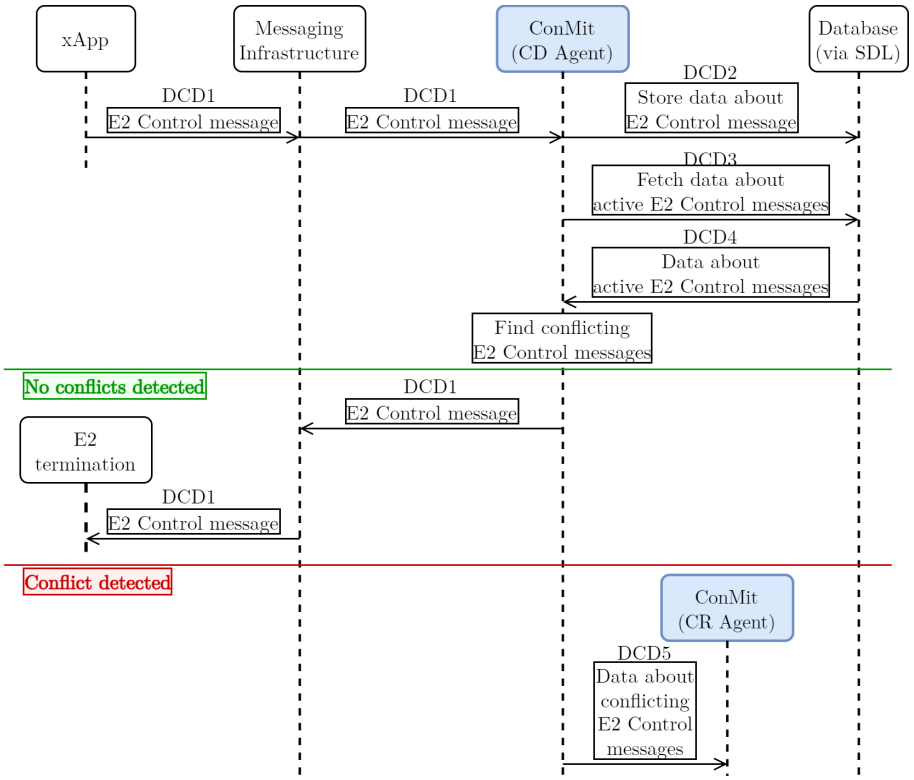

FIGURE 5.10: Proposed Direct Conflict Detection procedure in CMF

Source: own elaboration

LISTING 5.1: DCD1 - E2 Control message

```
{
  "source": "xApp1",
  "command": "Modify",
  "targetParameter": "TxPower",
  "targetValue": "32",
  "targetCell": "000-000-000-000001",
  "timestamp": "Wed Jan 01 00:00:00.000 2020",
  "controlTimespan": "500"
}
```

Upon interception, the CD Agent logs this control decision in the RIC's database to ensure the system is aware of the pending change. It issues a Store command to the "RecentlyChangedParameters" table, encapsulating the original control content, as depicted in Listing 5.2.

Once the new decision is logged, the CD Agent must assess the current context to check for any conflicts. It sends a request (Listing 5.3) to the database to fetch all currently active control decisions recorded in the "RecentlyChangedParameters" table.

The database responds with a list of records (Listing 5.4). In this example scenario, the records reveal a conflict: xApp2 has an active decision modifying TxPower to 34, while the new decision from xApp1 attempts to set it to 32 on the same target cell.

The CD Agent logic identifies that both decisions target TxPower on cell 000-000-000-000001. Consequently, it sends a conflict report to the CR Agent. As shown in Listing 5.5, this Notify message specifies the conflict type as "direct" and includes the details of both conflicting decisions, allowing the CR Agent to determine the optimal resolution.

Listing 5.2: DCD2 - Store data about E2 Control message

```
{
  "command": "Store",
  "table": "RecentlyChangedParameters",
  "recordId": "1",
  "content": {
    "source": "xApp1",
    "command": "Modify",
    "targetParameter": "TxPower",
    "targetValue": "32",
    "targetCell": "000-000-000-000001",
    "timestamp": "Wed Jan 01 00:00:00.000 2020",
    "controlTimespan": "500"
  }
}
```

Listing 5.3: DCD3 - Fetch data about active E2 Control messages

```
{
  "command": "FetchAll",
  "table": "RecentlyChangedParameters"
}
```

Indirect Conflict Detection

Indirect conflicts cannot be directly observed pre-action by simply comparing parameter names impacted by control decisions. Therefore, the ICD procedure of CMF is designed to anticipate the conflicts by using knowledge of Parameter Groups (PGs). These groups capture parameters known to influence the same area of RAN operation. These PGs can be configured manually by the MNO or predefined in the standards. They could also be learned dynamically during network operation, for example, by the CD Agent performing correlation and pattern recognition using historical data and KPI anomaly reports from the PMon component. The PG configuration is stored in the Database ("Parameter Group definitions" in Figure 5.9). Additionally, the Database stores information about active control messages that modify any parameters within these groups ("Recently changed Parameter Groups" in Figure 5.9). These entries contain the same scope of data as "Recently changed parameters" related to DCD, but refer to PGs instead of specific control parameters.

The ICD detection logic is a slight modification of the DCD procedure. Note that the procedure for ICD does not include logging the E2 Control message into the RIC's database, as this is already done in DCD. ICD analyzes each new control decision but first maps its target parameter(s) to any applicable predefined PGs. After mapping the decision, ICD checks the entries in the "Recently changed Parameter Groups" table to find any conflicting decisions that are mapped to at least one of the same PGs. If a match is found, information about the conflicting control decisions is provided to the CR Agent. The conflict detection logic and related message flow for ICD are shown in Figure 5.11. An example information exchange for the proposed ICD procedure is presented in the following paragraphs.

Similar to DCD, the ICD procedure is triggered when an xApp issues an E2 control message (Listing 5.6).

To detect indirect conflicts, the CD Agent must first map the target parameter of the incoming message to relevant parameter groups. It issues a FetchAll command for the "ParameterGroups" table (Listing 5.7).

Listing 5.4: DCD4 - Data about active E2 Control messages

```
{
  "table": "RecentlyChangedParameters",
  "timestamp": "Wed Jan 01 00:00:00.040 2019",
  "records": [
    {
      "recordId": "0",
      "content": {
        "source": "xApp2",
        "command": "Modify",
        "targetParameter": "TxPower",
        "targetValue": "34",
        "targetCell": "000-000-000-000001",
        "timestamp": "Tue Dec 31 23:59:59.900 2019",
        "controlTimespan": "500"
      }
    },
    {
      "recordId": "1",
      "content": {
        "source": "xApp1",
        "command": "Modify",
        "targetParameter": "TxPower",
        "targetValue": "32",
        "targetCell": "000-000-000-000001",
        "timestamp": "Wed Jan 01 00:00:00.000 2020",
        "controlTimespan": "500"
      }
    }
  ]
}
```

The database responds with the definitions (Listing 5.8). Here, the TxPower parameter is shown to belong to two groups: "CellAffectQoS" and "CellAffectHandoverBoundary". Crucially, the "CellAffectHandoverBoundary" group also contains ElectricalTilt and HOMeasurementOffset.

With this context established, the CD Agent logs the group-level impact of the new decision into the "RecentlyChangedParameterGroups" table. Listing 5.9 shows the store command, which explicitly links the change in TxPower to the affected groups.

Next, the CD Agent queries the database for other active decisions that affect these specific groups (Listing 5.10).

The database returns the active group changes (Listing 5.11). The response reveals that while xApp1 is modifying TxPower, xApp2 has recently modified HOMeasurementOffset. Since both map to the shared "CellAffectHandoverBoundary" group, an indirect conflict is detected.

Consequently, the CD Agent triggers a conflict notification (Listing 5.12) to the CR Agent. The message identifies the conflict type as "indirect" and cites the specific group name "CellAffectHandoverBoundary", along with the two decisions causing the friction.

Implicit Conflict Detection

Implicit conflicts cannot be observed directly, and the mutual influence between conflicting decisions is not easily deduced. Therefore, an approach similar to DCD and ICD is not suitable for ImCD, which can only work reactively. This procedure relies on an additional component within the ConMit component to analyze PM data reported by RAN nodes and KPI degradation; this role is fulfilled by the proposed PMon component within the CMF. The detection of such

Listing 5.5: DCD5 - Data about conflicting E2 Control messages

```
{
  "source": "DCD",
  "command": "Notify",
  "conflictType": "direct",
  "timestamp": "Wed Jan 01 00:00:00.060 2020",
  "decisions": [
    {
      "content": {
        "source": "xApp2",
        "command": "Modify",
        "targetParameter": "TxPower",
        "targetValue": "34",
        "targetCell": "000-000-000-000001",
        "timestamp": "Tue Dec 31 23:59:59.900 2019",
        "controlTimespan": "500"
      }
    },
    {
      "content": {
        "source": "xApp1",
        "command": "Modify",
        "targetParameter": "TxPower",
        "targetValue": "32",
        "targetCell": "000-000-000-000001",
        "timestamp": "Wed Jan 01 00:00:00.000 2020",
        "controlTimespan": "500"
      }
    }
  ]
}
```

Listing 5.6: ICD1 - E2 Control message

```
{
  "source": "xApp1",
  "command": "Modify",
  "targetParameter": "TxPower",
  "targetValue": "32",
  "targetCell": "000-000-000-000001",
  "timestamp": "Wed Jan 01 00:00:00.000 2020",
  "controlTimespan": "500"
}
```

Listing 5.7: ICD2 - Fetch data about Parameter Group definitions

```
{
  "command": "FetchAll",
  "table": "ParameterGroups"
}
```

network performance anomalies can be achieved with either traditional statistical-based solutions or AI/ML-based approaches (such as the one by Casas et al. [156]). More considerations about such approaches for detection of conflict are included in Section 6.1.

The ImCD procedure is not triggered by any control decision (as DCD and ICD are) but by the PMon component signaling a KPI degradation. Once this trigger is provided, the CD Agent, executing the ImCD procedure, analyzes which parameters and/or PGs have been recently modified

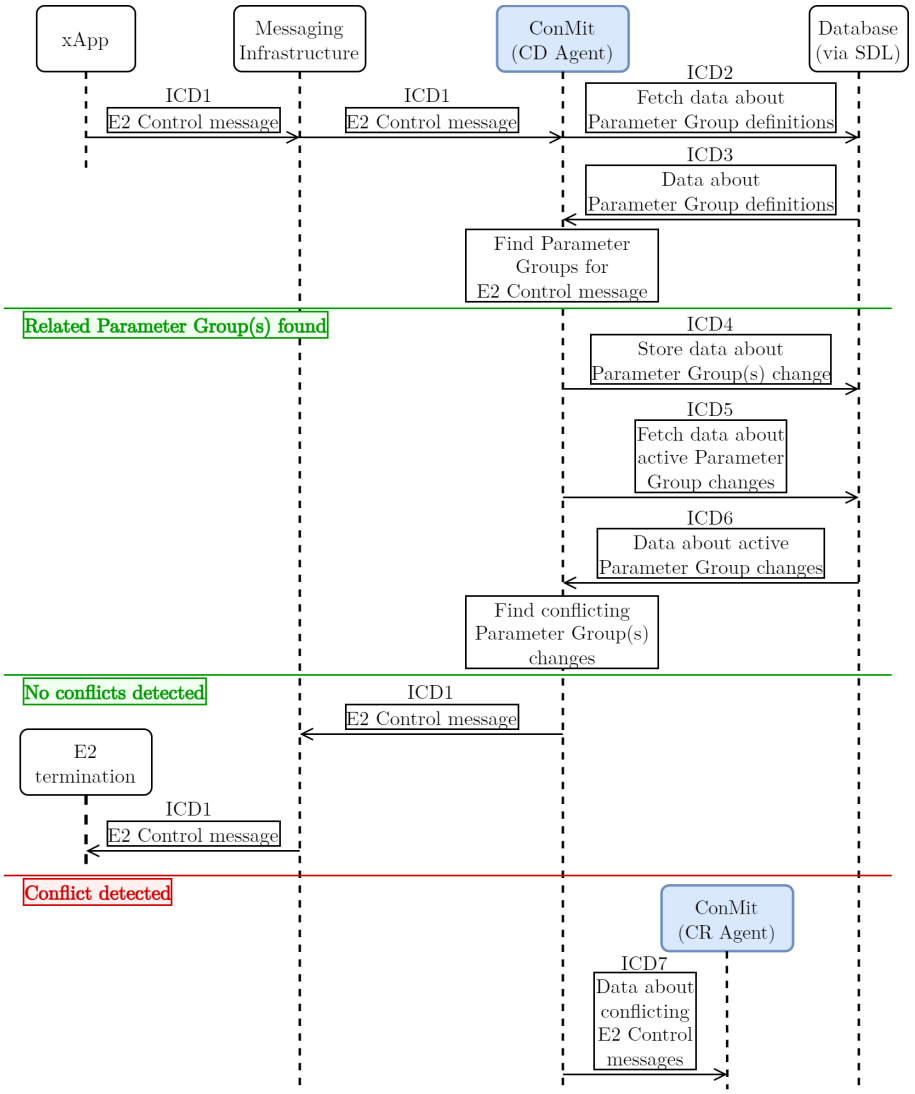

FIGURE 5.11: Proposed Indirect Conflict Detection procedure in CMF

Source: own elaboration

by multiple control decisions. The CD Agent utilizes the data captured in the RIC's database by DCD and ICD, so it does not need to monitor any control messages by itself.

If the CD Agent finds a correlation between any recent control decisions and the observed KPI degradation, it notes this event and increments internal counters associated with these control decisions. When any tracked counter breaches a predefined threshold (signaling a repeated pattern), the CD Agent provides data about conflicting applications' control decisions to the CR Agent. At this point in the procedure, the CD Agent may remove data about KPI degradation occurrences related to the reported conflict from the database.

A significant difference between DCD/ICD and ImCD is that the latter works post-action and, therefore, cannot entirely prevent conflicting RAN control messages from influencing the network. As such, it must be expected that implicit conflicts will have some negative impact on the network operation before they are mitigated. Similar to the DCD and ICD procedures described earlier, the message exchange flow and conflict detection logic for the ImCD procedure are depicted in Figure 5.12. An example information exchange for the proposed ImCD procedure is presented in the following paragraphs.

Listing 5.8: ICD3 - Data about Parameter Group definitions


```
{
  "table": "ParameterGroups",
  "timestamp": "Wed Jan 01 00:00:00.040 2020",
  "records": [
    {
      "recordId": "0",
      "content": {
        "groupName": "CellAffectQoS",
        "parameters": [
          {
            "changeTarget": "Cell",
            "parameterName": "TxPower"
          },
          {
            "changeTarget": "Cell",
            "parameterName": "ElectricalTilt"
          }
        ]
      }
    },
    {
      "recordId": "1",
      "content": {
        "groupName": "CellAffectHandoverBoundary",
        "parameters": [
          {
            "changeTarget": "Cell",
            "parameterName": "ElectricalTilt"
          },
          {
            "changeTarget": "Cell",
            "parameterName": "HOMeasurementOffset"
          },
          {
            "changeTarget": "Cell",
            "parameterName": "TxPower"
          }
        ]
      }
    }
  ]
}
```


The procedure begins when the PMon component receives performance data, as shown in Listing 5.13, indicating a specific event (e.g., AverageDLThroughputMbps at 12 Mbps).

PMon identifies this as a performance anomaly and logs it in the database via the Store command to the "RANKPIDegradationOccurences" table (Listing 5.14).

Once the degradation is logged, the CD Agent initiates an investigation to find a correlation between recent control decisions and the observed anomaly. It fetches the recent control history by querying both the "RecentlyChangedParameters" (Listing 5.15) and "ParameterGroups" (Listing 5.17) tables.

The database provides the recent raw parameter changes (Listing 5.16), showing recent modifications to TxPower by both xApp2 and xApp1.

Simultaneously, the agent fetches the group definitions (Listing 5.17) to understand the relationships between these parameters.

Listing 5.9: ICD4 - Store data about Parameter Group(s) change

```
{
  "command": "Store",
  "table": "RecentlyChangedParameterGroups",
  "content": {
    "changedParameterGroups": [
      {
        "groupName": "CellAffectQoS",
        "message": {
          "source": "xApp1",
          "command": "Modify",
          "targetParameter": "TxPower",
          "targetValue": "32",
          "targetCell": "000-000-000-000001",
          "timestamp": "Wed Jan 01 00:00:00.000 2020",
          "controlTimespan": "500"
        }
      },
      {
        "groupName": "CellAffectHandoverBoundary",
        "message": {
          "source": "xApp1",
          "command": "Modify",
          "targetParameter": "TxPower",
          "targetValue": "32",
          "targetCell": "000-000-000-000001",
          "timestamp": "Wed Jan 01 00:00:00.000 2020",
          "controlTimespan": "500"
        }
      }
    ]
  }
}
```

Listing 5.10: ICD5 - Fetch data about active Parameter Group changes

```
{
  "command": "FetchAll",
  "table": "RecentlyChangedParameterGroups"
}
```

The group definitions returned (Listing 5.18) confirm that parameters TxPower and ElectricalTilt are related via the "CellAffectQoS" group.

If the CD Agent finds a correlation (for example, a degradation consistently following specific changes to TxPower and HOMeasurementOffset), it logs this specific correlation instance into the "RANKPICorrDegradationOccurences" table (Listing 5.19).

To determine if this is a repeated pattern worthy of mitigation, the CD Agent fetches the history of these correlated degradations (Listing 5.20).

The database returns the historical records (Listing 5.21), showing multiple instances where this specific pair of decisions preceded a KPI drop.

If the number of occurrences breaches a predefined threshold, the CD Agent confirms an implicit conflict. It then sends a notification to the CR Agent (Listing 5.22), detailing the "implicit" conflict type, the affected KPI (AverageDLThroughput), and the history of degradation occurrences.

Listing 5.11: ICD6 - Data about active Parameter Group changes


```
{
  "table": "RecentlyChangedParameterGroups",
  "timestamp": "Wed Jan 01 00:00:00.080 2020",
  "records": [
    {
      "recordId": "0",
      "content": {
        "groupName": "CellAffectQoS",
        "messages": [
          {
            "source": "xApp1",
            "command": "Modify",
            "targetParameter": "TxPower",
            "targetValue": "32",
            "targetCell": "000-000-000-000001",
            "timestamp": "Wed Jan 01 00:00:00.000 2020",
            "controlTimespan": "500"
          }
        ]
      }
    },
    {
      "recordId": "1",
      "content": {
        "groupName": "CellAffectHandoverBoundary",
        "messages": [
          {
            "source": "xApp1",
            "command": "Modify",
            "targetParameter": "TxPower",
            "targetValue": "32",
            "targetCell": "000-000-000-000001",
            "timestamp": "Wed Jan 01 00:00:00.000 2020",
            "controlTimespan": "500"
          },
          {
            "source": "xApp2",
            "command": "Modify",
            "targetParameter": "HOMeasurementOffset",
            "targetValue": "2",
            "targetCell": "000-000-000-000001",
            "timestamp": "Tue Dec 31 23:59:59.890 2019",
            "controlTimespan": "500"
          }
        ]
      }
    }
  ]
}
```


5.4.4 Proposed methodology for evaluation of conflict resolution methods

As highlighted in Section 3.2.4, establishing methods for evaluation of ConMit solutions is crucial for quantifying and comparing the effectiveness of various detection and resolution approaches. This section describes a methodology proposed by the author for evaluation of CR methods.

To ensure a fair performance assessment, all methods must be tested in the same environment, with identical starting conditions and network configuration (apart from the CR setting). Furthermore, to minimize bias from any single scenario (e.g., where UEs are clustered in a specific

Listing 5.12: ICD7 - Data about conflicting E2 Control messages


```
{
  "source": "ICD",
  "command": "Notify",
  "conflictType": "indirect",
  "timestamp": "Wed Jan 01 00:00:00.086 2020",
  "groupName": "CellAffectHandoverBoundary",
  "decisions": [
    {
      "content": {
        "source": "xApp2",
        "command": "Modify",
        "targetParameter": "HOMeasurementOffset",
        "targetValue": "2",
        "targetCell": "000-000-000-000001",
        "timestamp": "Tue Dec 31 23:59:59.890 2019",
        "controlTimespan": "500"
      }
    },
    {
      "content": {
        "source": "xApp1",
        "command": "Modify",
        "targetParameter": "TxPower",
        "targetValue": "32",
        "targetCell": "000-000-000-000001",
        "timestamp": "Wed Jan 01 00:00:00.000 2020",
        "controlTimespan": "500"
      }
    }
  ]
}
```


location), the evaluation must consist of multiple runs with randomized factors, such as initial UE positions, UE movement, and network traffic distribution.

In summary, this methodology involves conducting multiple simulation runs with randomized user distributions. For each run, a weighted penalty score is calculated based on observed network faults. These scores are then normalized relative to the best performing method in that specific run, and averaged across all runs to produce a final, robust ranking of the CR methods.

The proposed methodology evaluates the CR methods by comparing them against a common baseline in a fixed environment. The author considers two baseline methods: no ConMit (no mitigation of conflicts) and CR with basic rule-based methods described in Section 5.4.5 (prioritization and prioritization with cooldown). This comparison makes it possible to assess whether the evaluated method surpasses the basic approaches. A good-performing CR method must perform better than (or at least no worse than) the "no ConMit" baseline in all considered cases. Ideally, it should also outperform the basic rule-based methods in some or all scenarios.

The author proposes a novel evaluation metric to quantify the impact of negative network events observed during the evaluation runs. The metric is a weighted sum of the counts of these negative events, where each count is scaled by a weight specific to the event type. The chosen events must be representative of the issues caused by the evaluated conflict. As such, the metric functions as a penalty score for the CR methods; a lower score is better.

As the author's work focused on conflicts between MRO and MLB xApps (a conflict involving applications for handover optimization), the proposed metric in this case utilizes network statistics relevant to handover performance. In this context, the negative network events are identified as:

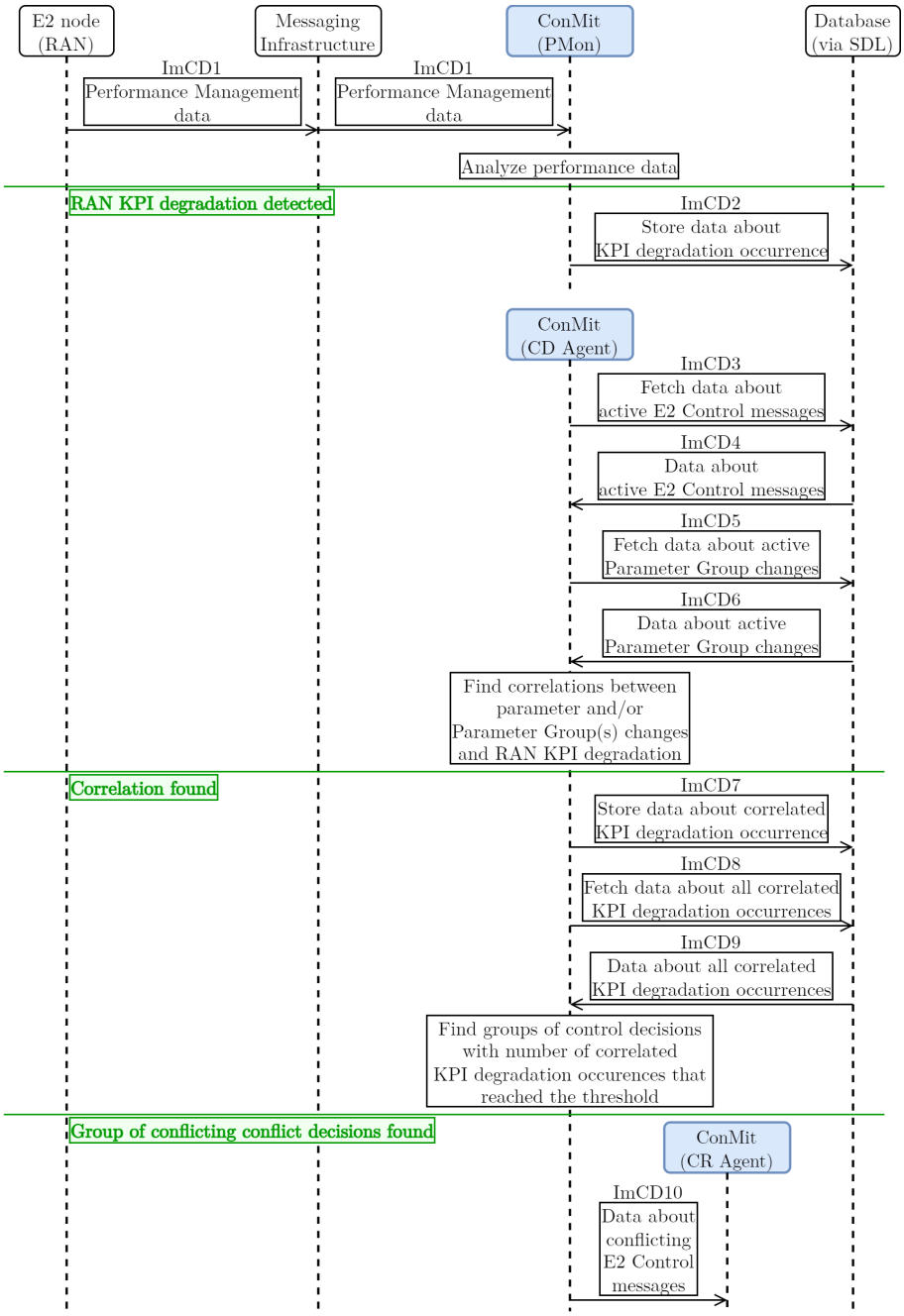

FIGURE 5.12: Proposed Implicit Conflict Detection procedure in CMF

Source: own elaboration

RLFs, ping-pong handovers, and CBs. As other use cases may require a different set of statistics used in the evaluation, the proposed method can be adapted as needed.

The formula for the proposed evaluation metric, adapted to the case of conflict between MLB and MRO applications, is depicted in (5.1).

$$P^i = w_{\mathrm{PP}}^{\mathrm{pen}}\, C_{\mathrm{PP}}^i + w_{\mathrm{RLF}}^{\mathrm{pen}}\, C_{\mathrm{RLF}}^i + w_{\mathrm{CB}}^{\mathrm{pen}}\, C_{\mathrm{CB}}^i. \tag{5.1}$$

The penalty weights are defined as parameters $w_{\mathrm{PP}}^{\mathrm{pen}}$, $w_{\mathrm{RLF}}^{\mathrm{pen}}$, and $w_{\mathrm{CB}}^{\mathrm{pen}}$. It is important to note that these are unit penalty values representing the severity of a single event, rather than a probability distribution; therefore, they do not sum to 1.0. For the evaluation in this work, the following values were selected to reflect the higher severity of service-denying events: $w_{\mathrm{PP}}^{\mathrm{pen}} = 0.05$,

Listing 5.13: ImCD1 - Performance Management data

```
{
  "source": {
    "name": "CU-UP1",
    "id": "000-000-000"
  },
  "event": {
    "id": "Perf00000001",
    "timestamp": "Tue Dec 31 23:59:59.890 2019",
    "type": "perfStream",
    "data": {
      "attributes": [
        {
          "name": "AverageDLThroughputMbps",
          "value": "12"
        },
        {
          "name": "AverageULThroughputMbps",
          "value": "5"
        }
      ]
    }
  }
}
```

Listing 5.14: ImCD2 - Store data about KPI degradation occurrence

```
{
  "command": "Store",
  "table": "RANKPIDegradationOccurences",
  "content": {
    "KPI": "AverageDLThroughput",
        "anomaly": "decreased",
    "timestamp": "Tue Dec 31 23:59:59.890 2019"
  }
}
```

Listing 5.15: ImCD3 - Fetch data about active E2 Control messages

```
{
  "command": "FetchAll",
  "table": "RecentlyChangedParameters"
}
```

$w_{\text{RLF}}^{\text{pen}} = 0.40$, and $w_{\text{CB}}^{\text{pen}} = 0.40$. The weights were chosen to highlight the difference in severity: ping-pong handovers are wasteful in terms of network resources, but do not cause drop of service, whereas RLFs and CBs are more severe, as they are directly responsible for service disruption and degradation of user experience. These parameters can be adapted by the operator to reflect different QoS priorities.

To compare results across different randomized runs, the absolute penalty scores (P^i) are normalized within the context of each evaluation run in relation to the lowest (best) penalty score from that run (P^{best}), as shown in (5.2) and (5.3). In this relative comparison, the best-performing method for a give run scores 100%, and all other methods score 100% or more.

$$P^{\text{best}} = \min_i \{P^i\} \tag{5.2}$$

Listing 5.16: ImCD4 - Data about active E2 Control messages

```
{
  "table": "RecentlyChangedParameters",
  "timestamp": "Wed Jan 01 00:00:00.040 2019",
  "records": [
    {
      "recordId": "0",
      "content": {
        "source": "xApp2",
        "command": "Modify",
        "targetParameter": "TxPower",
        "targetValue": "34",
        "targetCell": "000-000-000-000001",
        "timestamp": "Tue Dec 31 23:59:59.900 2019",
        "controlTimespan": "500"
      }
    },
    {
      "recordId": "1",
      "content": {
        "source": "xApp1",
        "command": "Modify",
        "targetParameter": "TxPower",
        "targetValue": "32",
        "targetCell": "000-000-000-000001",
        "timestamp": "Wed Jan 01 00:00:00.000 2020",
        "controlTimespan": "500"
      }
    }
  ]
}
```

Listing 5.17: ImCD5 - Fetch data about active Parameter Group changes

```
{
  "command": "FetchAll",
  "table": "ParameterGroups"
}
```

$$P^i_{\text{rel}} = \frac{P^i}{P^{\text{best}}} \cdot 100\% \tag{5.3}$$

Finally, to get a single score for each CR method, its relative penalty scores (P^i_{rel}) are averaged across all evaluation runs, each having a different, randomized scenario. The methods are then ranked based on this final average score: the lower the average relative penalty score, the better a method is.

5.4.5 Proposed rule-based conflict resolution methods

Conflict resolution in the CMF is designed to be highly flexible, allowing operators to implement the CR methods of their choice. To evaluate the proposed framework and establish a performance baseline, the author developed two basic rule-based resolution algorithms described in the following paragraphs.

The author's initial article on the CMF [2] introduced a simple *prioritization* method. In this approach, the CR Agent resolves conflicts by always prioritizing the control decisions of higher-ranking applications over those with lower rankings. As a result, the lower-ranking applications'

Listing 5.18: ImCD6 - Data about active Parameter Group changes

```
{
  "table": "ParameterGroups",
  "timestamp": "Wed Jan 01 00:00:00.040 2020",
  "records": [
    {
      "recordId": "0",
      "content": {
        "groupName": "CellAffectQoS",
        "parameters": [
          {
            "changeTarget": "Cell",
            "parameterName": "TxPower"
          },
          {
            "changeTarget": "Cell",
            "parameterName": "ElectricalTilt"
          }
        ]
      }
    },
    {
      "recordId": "1",
      "content": {
        "groupName": "CellAffectHandoverBoundary",
        "parameters": [
          {
            "changeTarget": "Cell",
            "parameterName": "ElectricalTilt"
          },
          {
            "changeTarget": "Cell",
            "parameterName": "HOMeasurementOffset"
          },
          {
            "changeTarget": "Cell",
            "parameterName": "TxPower"
          }
        ]
      }
    }
  ]
}
```

decisions are rejected and do not have any effect in the network. An application's priority level should be selected based on its criticality. The specific priority order can be either configured statically, where it does not change unless manually modified by the MNO, or modified dynamically based on current policies.

The *prioritization with cooldown* method, introduced in a subsequent work by the author focusing on mitigation of indirect conflicts [5], expands on the prioritization concept. As with the basic method, one of the applications involved in the conflict is given priority over the others. The key difference is that after rejecting the conflicting control decisions from the non-prioritized applications, the CMF also blocks all subsequent decisions from these applications for that specific control target for a limited *cooldown period*. During this time, the non-prioritized applications remain free to provide control decisions for other, unaffected control targets. This approach allows the prioritized application's decision to take effect and optimize the target without interference from the other applications' decisions. Once the cooldown period expires, the muted applications' control

Listing 5.19: ImCD7 - Store data about correlated KPI degradation occurrence

```
{
  "command": "Store",
  "table": "RANKPICorrDegradationOccurences",
  "content": {
    "KPI": "AverageDLThroughput",
        "anomaly": "decreased",
    "timestamp": "Tue Dec 31 23:59:59.890 2019",
    "message": {
      "conflictingDecisions": [
        {
          "source": "xApp1",
          "command": "Modify",
          "targetParameter": "TxPower",
          "targetValue": "32",
          "targetCell": "000-000-000-000001",
          "timestamp": "Tue Dec 31 23:59:59.800 2019",
          "controlTimespan": "500"
        },
        {
          "source": "xApp2",
          "command": "Modify",
          "targetParameter": "HOMeasurementOffset",
          "targetValue": "2",
          "targetCell": "000-000-000-000022",
          "timestamp": "Tue Dec 31 23:59:59.670 2019",
          "controlTimespan": "650"
        }
      ]
    }
  }
}
```

Listing 5.20: ImCD8 - Fetch data about all correlated KPI degradation occurrences

```
{
  "command": "FetchAll",
  "table": "RANKPICorrDegradationOccurences"
}
```

decisions are processed normally again. The duration of the cooldown period can be configured statically by the MNO or set dynamically by the CR Agent based on the conflict's impact or configuration from the ConMit Supervisor component.

The author also designed a more robust AI/ML-based CR method, which aims to dynamically adapt its resolution actions to the conflict's characteristics and current network conditions. This proposed method is described in detail and evaluated against these rule-based baselines in Section 6.2.

5.4.6 Example event sequence in Conflict Mitigation Framework

To clearly present the sequence on events during ConMit using CMF, an example event sequence for ICD is presented in Figure 5.13. The sequence considers a case of indirect conflict between MRO and MLB applications, where a configured PG for cell parameters affecting handovers is used to detect the conflict and *prioritization of MRO* method is used to resolve it.

In the figure, in (1) we observe the initial state of the O-RAN environment, together with a visual representation of the configured PG "AffectHandover" containing a set of 5 control parameters.

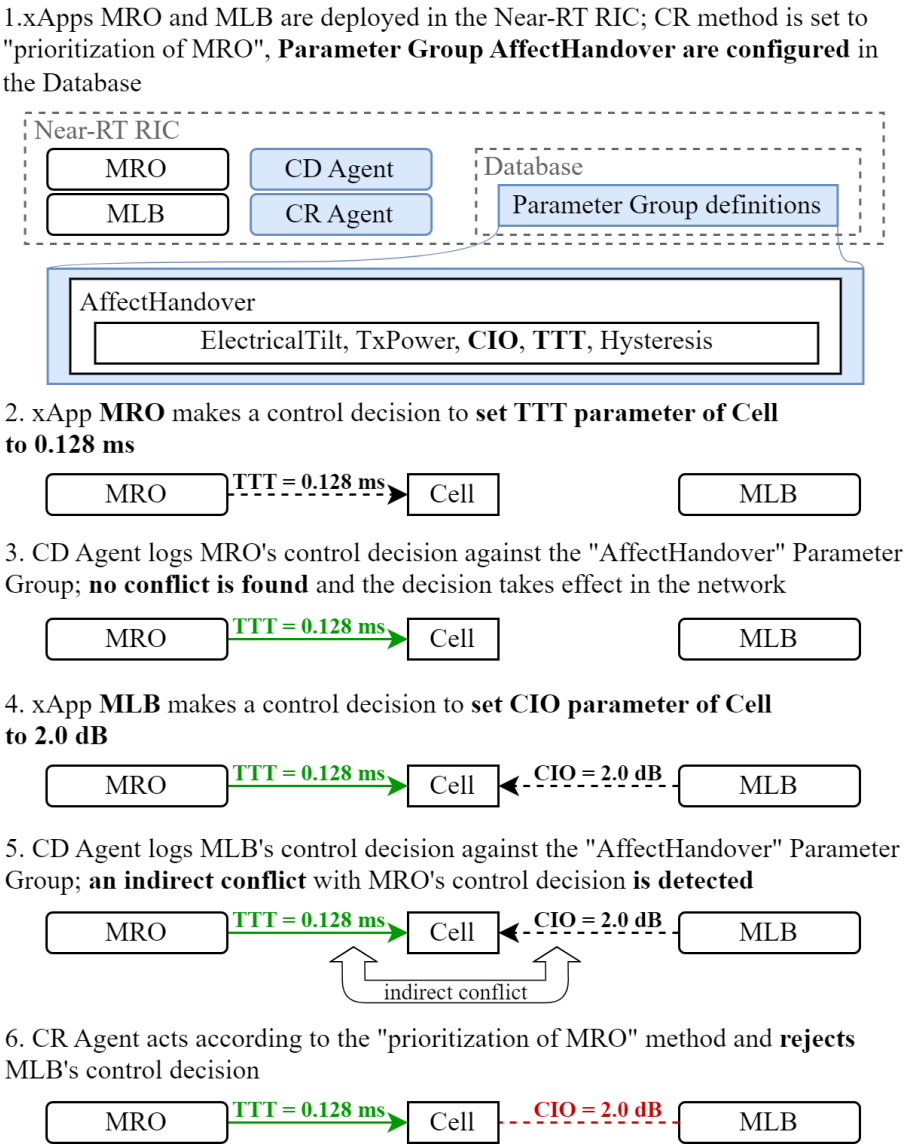

FIGURE 5.13: Example event sequence for ICD in CMF with prioritization CR method

Source: own elaboration

Then, in (2), the MRO xApp sets the cell's TTT to 0.128 ms. As a result, in (3), the CD Agent logs the control decisions in the RIC's database and determines there is no control conflict detected; as a result, the cell's TTT parameter is set to 0.128 ms, as per MRO's decision. In (4), MLB decides to set the CIO of the cell to 2.0 dB. Again, in (5), the CD Agent logs the new control decision and checks for conflicts. This time, due to the active control decision of MRO, the CD Agent detects the conflict and signals it to the CR Agent. Finally, in (6), the CR Agent decides to reject MLB's decision, according to its configured prioritization of MRO CR method.

5.4.7 Evaluation of the proposed rule-based conflict resolution methods

The two proposed rule-based CR methods are assessed to determine which one performs best and should be primarily used for evaluation of more robust methods. The methods tested according to the methodology described in Section 5.4.4 are:

- ConMit disabled,

Table 5.1: UE deployment configurations

Configuration	UEs in macro BS	UEs per micro BS	Total UE count
Small (S)	15	6	58
Medium (M)	20	8	77
Large (L)	30	10	101

- prioritization of MLB,
- prioritization of MRO,
- prioritization of MLB with cooldown,
- prioritization of MRO with cooldown.

The cases with prioritization with cooldown method are configured with a 10 second cooldown period.

To evaluate the efficiency of the CR methods, the simulator described in Chapter 4 is used. The simulation scenario comprises a hexagonal arrangement of eight 5G BSes. A single macro BS collocated with a micro BS is located in the center of the simulated area, with six evenly spaced micro BSes surrounding it. Each micro BS is configured with three 5G cells, each covering a 120-degree sector, providing full 360-degree coverage for each BS. The central macro BS additionally hosts three 5G cells shifted 180 degrees in relation to the micro cells. The macro cells provide enhanced coverage across the simulation area, enabling the network to balance load between micro and macro cells. The simulation area is defined as a circle with a radius of 450 m, representing the service area of the RAN, which UEs cannot cross. A graphical representation of the considered O-RAN network with cells numbered and simulation area boundary represented as a black outline is presented in Figure 5.14.

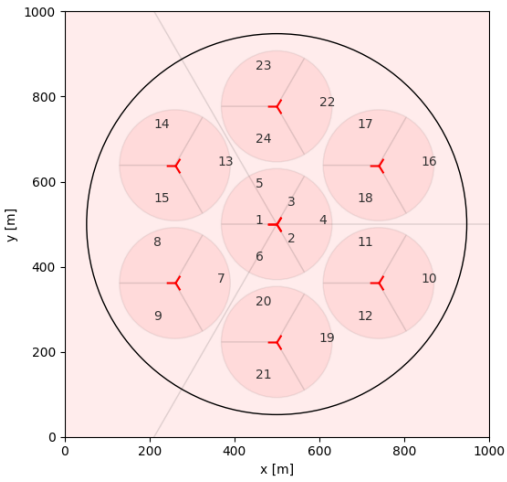

Figure 5.14: Graphical representation of the simulation area

Source: own elaboration

To simulate a dense user deployment, UEs are distributed within the simulation area. Three UE deployment configuration variants differing in number of UEs per each BS are considered as shown in Table 5.1. In addition, one extra UE is included on top of the per-BS configurations, the column with total UE count in the table includes this. Each UE travels within the simulated area, randomly changing the direction of its movement with a small probability. The default UE movement speed is $6\,\mathrm{m\,s^{-1}}$, but 25% of the UEs represent vehicles and travel at a higher speed of $30\,\mathrm{m\,s^{-1}}$, reflecting the diverse mobility of UEs in real-world RAN.

Table 5.2: User traffic profiles

Profile	Share of UEs (%)	Traffic type	Throughput demand
1	20	Low-bitrate voice	250 kbit s^{-1}
2	40	Medium-bitrate data	10 Mbit s^{-1}
3	40	High-bitrate data	50 Mbit s^{-1}

Table 5.3: Evaluation results for rule-based CR methods based on 16 simulation runs within each UE deployment scenario

Rank	CR method	Average penalty	Average relative penalty (%)	Delta to best (pp)
	Small UE deployment configuration			
1	**Prioritization of MRO with cooldown**	292.82	100.4	0.0
2	ConMit disabled	302.83	103.9	3.5
3	Prioritization of MLB	307.05	105.3	4.9
4	Prioritization of MRO	307.39	105.5	5.1
5	Prioritization of MLB with cooldown	333.84	114.5	14.1
	Medium UE deployment configuration			
1	**Prioritization of MRO with cooldown**	413.44	100.8	0.0
2	Prioritization of MRO	422.56	103.0	2.2
3	ConMit disabled	426.15	103.9	3.1
4	Prioritization of MLB	426.31	103.9	3.1
5	Prioritization of MLB with cooldown	445.13	108.5	7.7
	Large UE deployment configuration			
1	**Prioritization of MRO with cooldown**	570.38	100.3	0.0
2	ConMit disabled	583.15	102.6	2.3
3	Prioritization of MRO	584.03	102.7	2.4
4	Prioritization of MLB	586.42	103.2	2.9
5	Prioritization of MLB with cooldown	601.21	105.8	5.5

The simulated UEs are further characterized by one of three user traffic profiles shown in Table 5.2. This combination of user traffic profiles generates a heterogeneous traffic load and, therefore, challenges the network's ability to maintain high QoS across various cell and user categories. As such, the two xApps are in indirect conflict with each other.

The simulated network incorporates a Near-RT RIC hosting two xApps: MRO and MLB. The MRO xApp optimizes the cells' handover hysteresis and TTT with a goal of minimizing the number of ping-pong handover and RLFs. Simultaneously, the MLB xApp controls the CIO for each cell to balance the UE load across cells, promoting handovers from highly congested cells to those with fewer traffic.

To avoid bias from a specific single scenario, 16 simulation runs with randomized initial placement of UEs are conducted for each combination of tested CR method and UE deployment configuration. The total duration of each simulation run is set to 1000 seconds, but to ensure that the network operates under stable conditions before meaningful measurements are taken, the initial 200 seconds of each run are discarded.

The results are presented in Table 5.3, showing average absolute and relative penalty metrics for each method. Furthermore, the "Delta to best" column contains the difference in percentage points (pp) between each method's average relative penalty and that of the best-performing method in given UE deployment scenario.

Regardless of the UE deployment configuration, the *prioritization of MRO with cooldown* method is the clear best performer. Its advantage over the second-best method is confidently exceeding 2 *pp* in all cases. Moreover, it is the only CR method that consistently outperforms the *ConMit disabled* baseline across all UE deployment configurations. Other than this method, only *prioritization of MRO* in the medium UE deployment configuration managed to perform better than having no ConMit measures, by a small margin of less than 1 *pp*. As such, it can be concluded that the proposed methods, other than the *prioritization of MRO with cooldown*, do not offer a significant performance increase over disabled ConMit to justify their deployment.

The CR method that performs the worst a large margin is *prioritization of MLB with cooldown*, which trails the best-performing method by up to 14.1 *pp* of the average relative penalty metric in the worst observed case. In general, it can be observed that prioritization of MRO is better than prioritization of MLB. The only exception is in the small UE deployment configuration, where basic prioritization of MLB yields slightly better results than applying the same approach for MRO.

An important takeaway from the results is that a poorly suited CR method may negatively influence the network more than deploying no CR solution at all. Furthermore, the effectiveness of the prioritization methods is highly dependent on which application is prioritized. The presented results clearly illustrate that the *prioritization with cooldown* method is responsible for both the best (when prioritizing MRO) and the worst (when prioritizing MLB) results.

These results highlight a critical issue in ConMit operation through static rule-based approaches. While a well-suited rule (prioritization of MRO) yields stability, a misconfigured rule (prioritization of MLB) actively harms network performance by enforcing suboptimal decisions, leading to results worse than having no mitigation at all. This confirms that static rules lack the adaptability required for complex scenarios, motivating the need for the AI-driven approach, such as the one proposed by the author and discussed in the Section 6.2.

The results shown above are in line with the author's previous work, which also found that assigning a higher priority to MRO was preferable for both basic prioritization [2] and prioritization with cooldown [5] methods.

5.5 Supervision and adaptation

Supervision and adaptation of the ConMit operation is crucial for ensuring consistent effective mitigation of control conflicts in O-RAN. The logic deployed in the proposed ConMit components (i.e., CD Agent, CR Agent, and PMon) may lack the capability to dynamically adapt to changing network conditions or new A1 policies provided by the Non-RT RIC, hence, creating a requirement for such functionality. For that purpose, the S&A control loop provides closed-loop monitoring and reconfiguration of the ConMit-related components in the RIC. By monitoring performance metrics, it provides a continuous feedback loop to ensure that any conflict resolution actions remain effective without negative impact on the network operation.

The ConMit Supervisor, which is the primary actor responsible for executing the S&A activities for ConMit, monitors how well the ConMit components mitigate detected control conflicts, identifies potential improvements, and adjusts their operational parameters. This is done by reconfiguring of ConMit components to align with current policy set by the MNO, KPI trends, and general network conditions.

An example of the S&A control loop in action is the ConMit Supervisor receiving an alarm from PMon about KPI deterioration following CR Agent's resolution actions, and consequently deciding that to switch to a different CR method supported by the CR Agent. Additionally, for parameterized CR methods like prioritization and prioritization with cooldown, the ConMit Supervisor may choose to reconfigure the application priorities or duration of the cooldown period.

The evaluation of specific resolution actions taken by the CD&R control loop is expected to require complex logic within the ConMit Supervisor. One approach for this evaluation is what-if analysis. In this method, the ConMit Supervisor component estimates the hypothetical effects of alternative ConMit decisions and assess the optimality of the resolution applied by the ConMit components executing the CD&R control loop. Alternatively, supervision of ConMit may include analysis of the evaluation of the conflicts based on the outputs of the CD Agent, as described in Section 5.2.1.

An example of what-if analysis for O-RAN was recently published by Hou et al. [149]. As described previously in Section 3.4.3, their CCKE framework generates counterfactual KPIs estimations as a range of values, which, by design, is guaranteed to contain the true counterfactual KPI value. A ConMit Supervisor utilizing this method could perform this what-if analysis and determine whether the action taken by the CR Agent was optimal, or if a different ConMit configuration would have yielded a better outcome. Considering such analysis as a part of the process would help the ConMit Supervisor determine the best ConMit configuration for the current network conditions.

The S&A control loop envisioned as part of the proposed ConMit solutions is depicted in Figure 5.15.

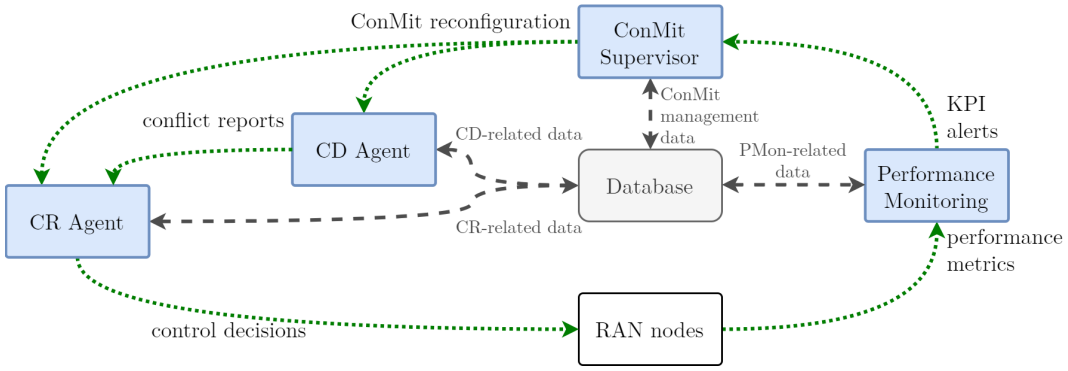

FIGURE 5.15: Supervision and adaptation control loop

Source: own elaboration

Listing 5.21: ImCD9 - Data about all correlated KPI degradation occurrences


```
{
  "table": "RANKPICorrDegradationOccurences",
  "timestamp": "Wed Jan 01 00:00:00.000 2020",
  "records": [
    {
      "recordId": "0",
      "content": {
        "KPI": "AverageDLThroughput",
                "anomaly": "decreased",
        "timestamp": "Tue Dec 31 23:59:59.890 2019",
        "conflictingDecisions": [
          {
            "source": "xApp1",
            "command": "Modify",
            "targetParameter": "TxPower",
            "targetValue": "32",
            "targetCell": "000-000-000-000001",
            "timestamp": "Tue Dec 31 23:59:59.800 2019",
            "controlTimespan": "500"
          },{
            "source": "xApp2",
            "command": "Modify",
            "targetParameter": "HOMeasurementOffset",
            "targetValue": "2",
            "targetCell": "000-000-000-000022",
            "timestamp": "Tue Dec 31 23:59:59.670 2019",
            "controlTimespan": "650"
          }
        ]
      }
    },{
      "recordId": "1",
      "content": {
        "KPI": "AverageDLThroughput",,
                "anomaly": "decreased",
        "timestamp": "Tue Dec 31 23:59:57.762 2019",
        "conflictingDecisions": [
          {
            "source": "xApp1",
            "command": "Modify",
            "targetParameter": "TxPower",
            "targetValue": "30",
            "targetCell": "000-000-000-000001",
            "timestamp": "Tue Dec 31 23:59:57.560 2019",
            "controlTimespan": "500"
          },{
            "source": "xApp2",
            "command": "Modify",
            "targetParameter": "HOMeasurementOffset",
            "targetValue": "3",
            "targetCell": "000-000-000-000022",
            "timestamp": "Tue Dec 31 23:59:57.020 2019",
            "controlTimespan": "800"
          }
        ]
      }
    }
  ]
}
```

Listing 5.22: ImCD10 - Data about conflicting E2 Control messages


```
{
  "source": "ImCD",
  "command": "Notify",
  "conflictType": "implicit",
  "timestamp": "Wed Jan 01 00:00:00.084 2020",
  "degradKPI": "AverageDLThroughput",
  "degradOccurrences": [
    {
      "timestamp": "Tue Dec 31 23:59:59.890 2019",
          "anomaly": "increased",
      "decisions": [
        {
          "source": "xApp1",
          "command": "Modify",
          "targetParameter": "TxPower",
          "targetValue": "32",
          "targetCell": "000-000-000-000001",
          "timestamp": "Tue Dec 31 23:59:59.800 2019",
          "controlTimespan": "500"
        },
        {
          "source": "xApp2",
          "command": "Modify",
          "targetParameter": "HOMeasurementOffset",
          "targetValue": "2",
          "targetCell": "000-000-000-000022",
          "timestamp": "Tue Dec 31 23:59:59.670 2019",
          "controlTimespan": "650"
        }
      ]
    },
    {
      "timestamp": "Tue Dec 31 23:59:57.762 2019",
          "anomaly": "increased",
      "decisions": [
        {
          "source": "xApp1",
          "command": "Modify",
          "targetParameter": "TxPower",
          "targetValue": "30",
          "targetCell": "000-000-000-000001",
          "timestamp": "Tue Dec 31 23:59:57.560 2019",
          "controlTimespan": "500"
        },
        {
          "source": "xApp2",
          "command": "Modify",
          "targetParameter": "HOMeasurementOffset",
          "targetValue": "3",
          "targetCell": "000-000-000-000022",
          "timestamp": "Tue Dec 31 23:59:57.020 2019",
          "controlTimespan": "800"
        }
      ]
    }
  ]
}
```

Chapter 6

Proposed AI/ML utility in O-RAN conflict mitigation

The O-RAN architecture is designed with native support for AI/ML techniques. These methods are ideally suited for ConMit, where the execution of complex processing logic against large datasets can address some of the challenges in O-RAN ConMit: ensuring the reliability of conflict detection mechanisms and determining optimal CR logic, as pointed out in Sections 3.2 and 3.4.3. General ideas of utility of such solutions in O-RAN ConMit, along with an implementation of an ANN-based CR method, are presented in the following sections.

6.1 AI/ML-aided conflict mitigation

The complexity and dynamic nature of O-RAN networks make efficient conflict mitigation using static, rule-based conventional approaches challenging. Techniques based on AI and ML are envisioned as essential enablers for robust ConMit measures, which can adapt to ever-changing network conditions and the evolving suite of deployed applications.

For CD methods, the utilization of AI/ML is expected to be the primary solution for detection of implicit conflicts. Specifically, a CD Agent could employ an AI model, such as an ANN, capable of forecasting KPI trends and detecting KPI anomalies based on monitored network metrics. Such insights about the network can be used to either predict a future conflict before it happens or to determine that an implicit conflict has happened. Beyond implicit conflicts, the detection of indirect conflicts will also benefit from AI/ML techniques. For instance, advanced neural networks like GNN can be used to dynamically build conflict graphs, capturing the dependencies between control parameters and performance metrics (as shown in [142, 146]). Information about these dependencies learned through the usage of GNN can then be used by the CD Agent to update its configuration of PGs, enabling automated maintenance of the CD configuration in a network with an evolving set of deployed applications.

AI/ML also enables the switch from conflict resolution using static rules (e.g., fixed application priorities) to dynamic optimization. The expected approach involves an AI model acting as the core of the resolution logic. This model ingests a wide range of parameters describing network state and the detected conflict to decide on the optimal set of resolution actions. Such a system can benefit from ML approaches, like RL, allowing the AI model to adapt its operation and improve its decision-making over time based on the reward signal from the O-RAN environment. This approach is explored further in the author's proposal for ANN-based CR method described in Section 6.2.

Last but not least, ConMit supervision and adaptation activities can also benefit greatly from AI/ML. Similarly to the proposed mechanism for implicit conflict detection, the ConMit Supervisor can ingest alerts from PMon to make informed decisions about reconfiguration of ConMit components. An AI model deployed within the ConMit Supervisor can make use of information from PMon to derive optimal parameters for the ConMit components and reconfigure them as the network conditions change. For example, upon a change of network conditions, it may decide that the conflicting applications are less temporarily dependent and, as a result, the duration of the cooldown period in the prioritization with cooldown CR method can be shortened.

6.2 ACCoRD: AI/ML-based conflict resolution

To address the identified gap in robust CR methods for ConMit in the Near-RT RIC, the author proposes ACCoRD, a novel ANN-based CR solution designed for deployment within the proposed Conflict Mitigation Framework. ACCoRD functions as the main decision-making entity within the CR Agent, executing the *Computation of Resolution Actions* function shown in Figure 5.3. By integrating an ANN, ACCoRD analyzes control conflicts by processing data regarding conflicting decisions and current network decisions to infer target resolution actions. The architecture of the Near-RT RIC with CMF components and ACCoRD is shown in Figure 6.1.

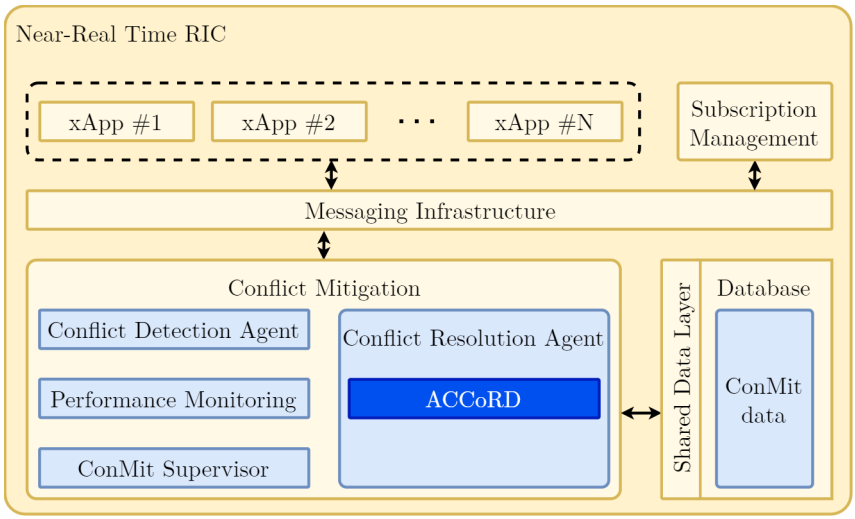

FIGURE 6.1: Near-RT RIC architecture with CMF components (light blue) and ACCoRD (blue)

Source: own elaboration

Functionally, ACCoRD operates by intercepting conflict reports from the CD Agent, collecting the necessary context information, and pre-processes inputs for its ANN. For each detected conflict, the ANN infers a set of resolution actions, which are applied to the conflicting control decisions. Such post-resolution decisions are then executed by the CR Agent. An actor-critic architecture trained via the PPO-Clip algorithm is employed to enable ANN's adaptation to dynamic network conditions. The specifics of the PPO-Clip algorithm and its application in this context are detailed in Section 6.2.4. This training process relies on a reward value computed within ACCoRD based on observed network metrics. The data flow for ACCoRD within the CMF is illustrated in Figure 6.2. Key information exchanges between components within the Near-RT RIC are labeled sequentially according to the order of events in the ConMit operation; messages internal to the ConMit component are not labeled. To maintain clarity, interactions extraneous to the core CR logic in ACCoRD (such as communication between CD Agent and PMon, or database integration) are omitted from the figure.

Operation of ACCoRD follows these steps (information exchanges from Figure 6.2 are mentioned by label where applicable):

I Control decision issuance: xApps submit their control decisions as E2 Control messages. These are transferred via the Messaging Infrastructure to the ConMit component, as required for CMF operation.

II Conflict detection: The messages are routed to the ConMit component for processing by the CD Agent:

   a) If no conflict is detected, the CD Agent forwards the unaltered control decisions toward the target RAN node via the Messaging Infrastructure (3a); affected control decisions go right into Execution of control decisions step.

   b) If a conflict is detected, the CD Agent generates a conflict report and sends it to the CR Agent; affected control decisions continue into Computation of policy logits step.

III Computation of policy logits: the CR Agent forwards the conflict report to ACCoRD for computation of resolution actions. ACCoRD preprocesses this data along with global features and feeds it through the ANN to compute CR action logits and a critic value.

IV Execution of conflict resolution actions: ACCoRD interprets the logits computed by the ANN to determine resolution actions, applies them to conflicting control decisions, and submits the resulting post-resolution control decisions to the CR Agent, which sends post-resolution control decisions as E2 Control messages back to the Messaging Infrastructure (3b).

V Execution of control decisions: unaltered and post-resolution control decisions are passed as E2 Control messages through E2 termination (4) towards the target RAN nodes and applied (5).

VI Post-resolution observation: after resolving a conflict, ACCoRD triggers a post-resolution observation of the network to calculate the reward.

VII ANN training with PPO-Clip: ACCoRD uses the computed reward together with CR policy action logits and critic value to compute training feedback for the ANN using PPO-Clip algorithm.

While the framework discussed in this chapter is proposed for the resolution of general conflicts in O-RAN, the specific implementation presented in the following sections focuses on a concrete use case to demonstrate the solution's viability. Specifically, ACCoRD is implemented and evaluated in the context of resolving indirect conflicts between MRO and MLB xApps, a common and significant challenge in RAN optimization. This concrete implementation serves as a proof of concept for the broader applicability of the proposed AI/ML-based approach.

The efficiency of an AI/ML-based solution for ConMit relies heavily on its design. Specifically for ACCoRD, performance depends on the input feature selection, the ANN architecture, the utilized ML algorithm, and the reward function used during training. The following sections detail the design choices enabling ACCoRD to compute effective CR actions.

6.2.1 ANN inputs

The input architecture for ACCoRD's ANN is designed to simultaneously process the aggregate environment state and the details of conflicting control decisions. The ANN accepts inputs divided into two primary categories: global features, which provide contextual information regarding the

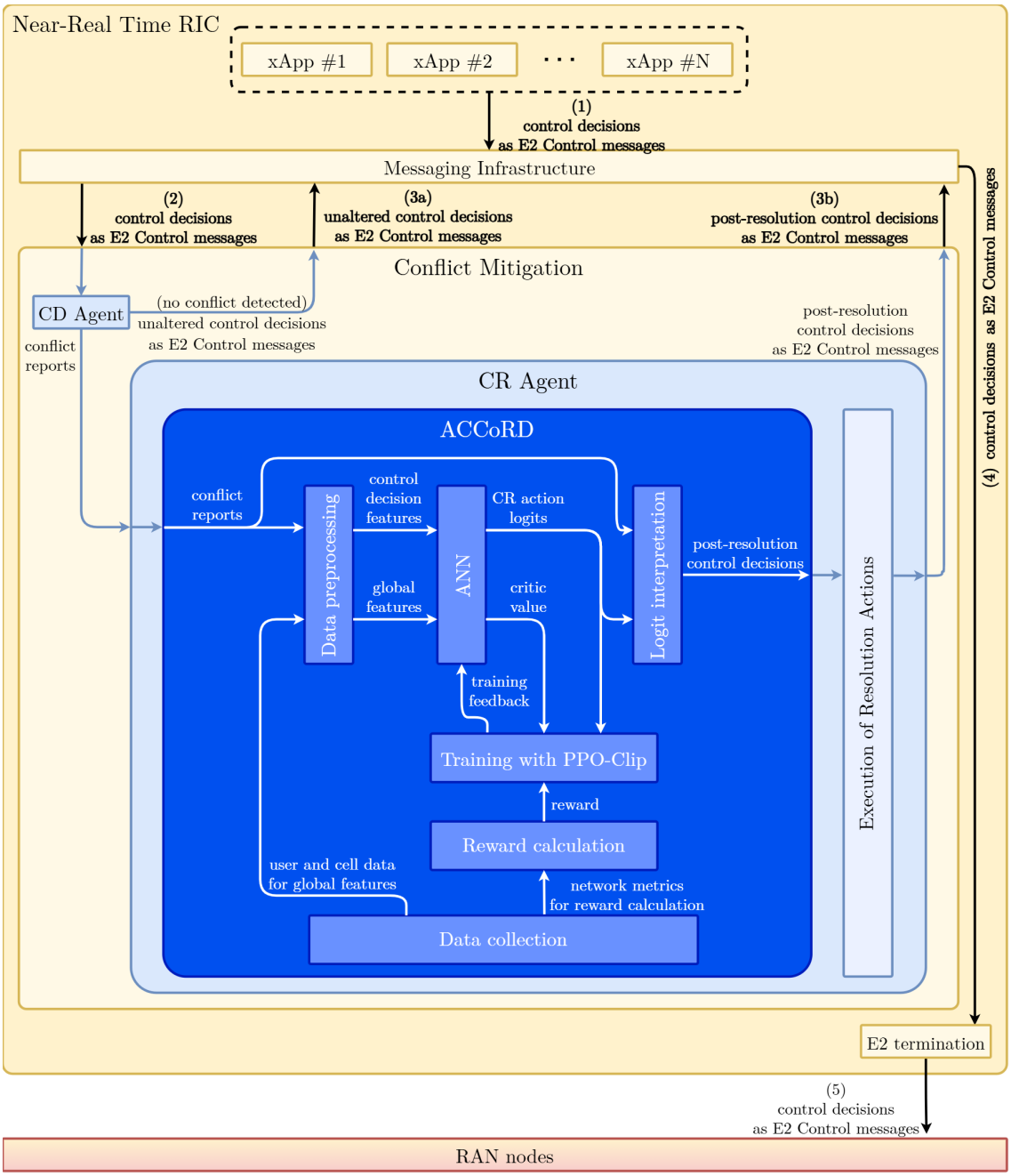

FIGURE 6.2: Data flow for ACCoRD within CMF

Source: own elaboration

target cell of the conflicting control decisions and its active users, and control decision features, which encode the specific control decisions involved in the conflict. These two sets of features are shown in Figures 6.2, 6.3 and 6.4. The specific input features of the ANN in ACCoRD are listed in Table 6.1.

Control decision features

Each inference step of the ANN processes input data describing up to N_{ConfDec} conflicting control decisions. To accommodate this variable number of inputs, the ANN ingests control decision features as a fixed-size, slot-based input tensor with dimensions $(N_{\text{ConfDec}}, 5)$. Here, N_{ConfDec} represents the maximum number of decision heads (slots) available in the network architecture.

Table 6.1: Input specification for ACCoRD's ANN

Category	Input feature	Description	Format
Control decision features	Parameter type embedding	One-hot vector identifying the parameter type (`NONE`, `CIO`, `TTT`, `HYST`).	Float tensor (4)
	Parameter value	Numeric value of the specific conflict parameter, normalized using min-max scaling for given parameter type.	Float scalar
Global features (cell state)	Hysteresis	Handover hysteresis configured in target cell, normalized by the maximum configurable hysteresis value.	Float scalar
	CIO	Cell Individual Offset configured in target cell, normalized by the maximum configurable CIO value.	Float scalar
	TTT	Time-to-trigger configured in target cell, normalized by the maximum configurable TTT value.	Float scalar
	Availability KPI	Percentage of time cell load is less than 100% value range: 0.0 to 1.0.	Float scalar
	HO Stability KPI	Percentage of non-ping-pong handovers, value range: 0.0 to 1.0.	Float scalar
	Throughput KPI	Average total cell throughput, normalized by 100 Mbps ($1e8$).	Float scalar
Global features (user state)	Movement speed	Average speed of users connected to target cell, normalized by 100 m/s (100).	Float scalar
	Requested bitrate	Average bitrate demanded by user profiles, normalized by 1 Gbps ($1e9$).	Float scalar
	Satisfaction KPI	Average user satisfaction ratio, ratio of assigned to requested bitrate (but not over 100%), value range: 0.0 to 1.0.	Float scalar
	Connection success KPI	Average ratio of successful connections, value range: 0.0 to 1.0.	Float scalar
	User throughput KPI	Average actual user throughput, normalized by 1 Mbps ($1e6$).	Float scalar

Each slot i corresponds to a specific decision head and contains a feature vector $\mathbf{x}_i$ of dimensionality 5. This vector is constructed by concatenating a parameter type embedding with the normalized parameter value:

$$\mathbf{x}_i = [\mathbf{e}_{type} \parallel v'_{param}]. \tag{6.1}$$

The components of the feature vector are defined as follows:

1. Parameter type embedding ($\mathbf{e}_{type}$): a fixed, non-trainable identity embedding; its dimensions are defined by $N_{\text{ConfDec}} + 1$ to allow for mapping of all possible control parameters during conflict and include a value representing an empty/padded decision. In the considered scenario, this embedding has 4 dimensions ($N_{\text{ConfDec}} = 3$, as described in Section 6.2.6). This one-hot encoding ensures the ANN interprets parameters based on learned categorical relationships rather than arbitrary numeric representations. The following four possible states map as follows:

   - `NONE` - $(1, 0, 0, 0)$ - empty/padded decision slot.
   - `CIO` - $(0, 1, 0, 0)$ - decision regarding the CIO parameter.
   - `TTT` - $(0, 0, 1, 0)$ - decision regarding the TTT parameter.
   - `HYST` - $(0, 0, 0, 1)$ - decision regarding the handover hysteresis parameter.

2. Parameter value (v'_{param}): a 1-dimensional scalar value of the parameter set by the conflicting control decision. To ensure input stability and improve training efficiency, this value is normalized to the range $[0, 1]$ using min-max scaling:

   $$v'_{param} = \frac{v_{param} - v_{min}}{v_{max} - v_{min}}, \tag{6.2}$$

   where v_{param} is the raw value from the control decision, and v_{min}, v_{max} are the configuration bounds specific to that parameter type (e.g., the valid range defined by the MLB xApp for CIO).

To simplify implementation, supported control parameter types are mapped to dedicated slots in the input data. If a specific parameter type is absent from a conflict (e.g., a conflict involving only TTT and handover hysteresis, missing CIO), its designated slot is filled with the `NONE` embedding of control parameter type and a zero value. Furthermore, an invalid slot mask is computed to exclude such empty slots from subsequent processing.

Global features

In addition to specific conflict parameters, the ANN ingests a set of global features describing the current state of the control target (the cell) and its active users ("global features" in Figures 6.2 and 6.3). These features anchor ACCoRD's decision-making in the current network context.

The global feature set consists of 11 floating-point values that summarize the dynamic state of the environment. This vector is constructed by concatenating cell-level and user-level data:

- Cell state (6 dimensions/parameters): this subset describes the state of the target cell for conflicting control decisions. These features include the current normalized values for the control parameters: hysteresis, CIO, and TTT. Additionally, it includes three KPIs: availability (percentage of time load is under 100%), handover stability (ratio of non-ping-pong handovers), and total throughput (normalized by a global maximum constant equal to 100 $Mbps$).
- User state (5 dimensions/parameters): this subset represents the aggregated state of all users connected to the target cell. The values are averaged across all relevant users to produce a fixed-size vector. The features include: movement speed, requested bitrate, satisfaction, connection success rate, and user throughput. If no users are connected to the target cell, this vector defaults all features to zero.

6.2.2 ANN outputs

The output layer of the ANN defines the action space of ACCoRD and provides the necessary auxiliary output for the actor-critic training algorithm. As such, the ANN generates three distinct sets of outputs: the action logits (actor values), a validity mask, and the critic value (state-value estimate). The action logits and critic value are shown in Figures 6.2, 6.3, and 6.4; the validity mask is omitted from the higher level illustrations in Figures 6.2 and 6.4 as it is considered a crucial part of the action logits. The specific outputs of the ANN in ACCoRD are listed in Table 6.2 and described in detail in the following sections.

Table 6.2: Output specification for ACCoRD's ANN

Category	Output tensor	Description	Format
Actor outputs	CR action logits	Raw, unnormalized action scores for every conflict slot representing the preference for each of the N_{A} defined CR actions.	Float tensor $(N_{\text{ConfDec}}, N_{\text{A}})$
	Validity mask	Boolean mask used to distinguish valid decision heads corresponding to real conflicts from padding slots.	Boolean tensor (N_{ConfDec})
Critic outputs	Critic value	Scalar estimate of the state value $V(s)$, representing the expected cumulative reward from the current state.	Float scalar

CR action logits (actor)

Since ACCoRD must simultaneously resolve multiple conflicting control decisions, the ANN architecture utilizes a multi-head output structure. The output layer consists of N_{ConfDec} dedicated decision heads, where each head corresponds to a specific conflict slot in the input tensor. Each decision head comprises a linear output layer with N_{A} neurons, where N_{A} represents the number of possible conflict resolution actions.

The raw output of each decision head is a vector of N_{A} logits–unnormalized values ranging from $-\infty$ to $+\infty$. In this context, a logit represents the ANN's preference to perform a specific CR action relative to others. A higher logit value indicates a stronger preference for that action. Mapping between logits and their respective CR actions is described in Section 6.2.2.

These logits serve as the input to a Softmax function, which transforms them into a valid probability distribution over the available CR actions. The probability $P(a_i)$ of selecting action a_i is given by (6.3).

$$P(a_i) = \frac{e^{l_i}}{\sum_{j=1}^{N_{\text{A}}} e^{l_j}}. \tag{6.3}$$

The application of 6.3 guarantees that the resulting output values strictly fall within the range $[0, 1]$ and sum up to exactly 1. This transformation is critical for numerical stability, as it converts unbounded logits into bounded probabilities, avoiding numerical overflow and providing the valid probabilistic interpretation of the resulting vector.

This probability distribution constitutes the stochastic CR policy $\pi_\theta(a|s)$ learned by the ANN, where θ represents the network weights parameterizing this policy. The optimization of these weights θ is achieved through the training process described in Section 6.2.4.

CR action space definition

The probability distribution $P(a_i)$ derived from the policy logits corresponds to a discrete action space $\mathcal{A}$. A discrete space was selected over a continuous one to balance the granularity of control with the convergence speed of the RL algorithm. While a continuous or highly granular action space allows for fine-tuned parameter adjustments, it significantly increases the dimensionality of the learning problem, often leading to unstable training or failure to converge within a reasonable timeframe.

The output of each decision head in ACCoRD's ANN corresponds to one of four distinct modification options for a conflicting control decision:

- `NO_MODIFICATION` - the control decision is applied without any modifications,
- `REJECTION_WITH_COOLDOWN` - the control decision is rejected. Additionally, the source xApp is put on a cooldown of duration $t_{\text{CR}} = 10$s for the specific control target, preventing control parameter oscillation and further repeat of the same conflict,
- `INCREASE_1` - the control decision is applied, but the target parameter value is incremented by a single logical step in relation to the original requested value,
- `DECREASE_1` - the control decision is applied, but the target parameter value is decremented by a single logical step in relation to the original requested value.

This discrete action space was selected to balance flexibility with learning complexity. Empirical testing conducted during the design phase indicated that expanding the action space (e.g., including

Table 6.3: Supported parameter values for MLB and MRO xApps

Parameter	xApp	Supported values	Unit
Cell Individual Offset	MLB	0.0, 0.5, 1.0, 1.5, 2.0, 2.5	dB
Handover hysteresis	MRO	0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0, 7.5, 8.0, 8.5, 9.0, 9.5, 10.0	dB
time-to-trigger	MRO	0.064, 0.08, 0.1, 0.128, 0.16, 0.256, 0.32, 0.48, 0.512, 0.64, 1.024, 1.28, 2.56, 5.12	s

±2 step modifications or action of rejection without cooldown) increased computational complexity without yielding statistically significant improvements in the final CR performance.

Effectively, the current action space configuration enables ACCoRD to replicate the logic of the *prioritization with cooldown* method (identified as the best-performing rule-based approach in Section 5.4.7) through a combination of `NO_MODIFICATION` and `REJECTION_WITH_COOLDOWN`. Beyond this baseline capability, the inclusion of `INCREASE_1` and `DECREASE_1` actions allows the trained ANN to perform minor adjustments to control parameter values when determined to be optimal.

The magnitude of the `INCREASE_1` and `DECREASE_1` actions is not a fixed numerical constant; rather, it is context-dependent. ACCoRD maps these abstract actions to specific values based on the discrete configuration sets supported by the target xApp.

Functionally, a single step increment sets the parameter to the next higher valid value defined in the application's configuration interface, while a decrement sets it to the next lower valid value. This approach ensures that ACCoRD always enforces valid configurations that the underlying xApps and RAN nodes can accept. Boundary conditions are handled by clamping: if a modified value exceeds the highest or lowest supported value, the parameter remains at that boundary.

To validate this approach within the scope of this work, the ACCoRD implementation is tailored to the MLB and MRO conflict scenario described in Section 5.4.4. Consequently, the step handling logic interprets the actions according to the discrete value sets supported by these specific xApps, as detailed in Table 6.3.

This discrete value step approach for increment/decrement strategy provides a unified action space for all control parameters, regardless of their supported value distribution. For parameters with linear spacing (e.g., handover hysteresis and CIO), this effectively results in fixed-step modifications (e.g., 0.5 dB steps). However, for parameters with non-linear spacing (e.g., TTT), this method ensures that ACCoRD strictly adheres to valid configuration steps without requiring separate logic.

Validity mask

Because the ANN utilizes a fixed number of decision heads (N_{ConfDec}) to process a variable number of conflicting decisions, not all output heads produce valid data for every inference step. To address this, the ANN outputs a validity mask, which is a boolean tensor of dimension N_{ConfDec}. This mask is not visible in Figures 6.2, 6.3, and 6.4, as it is considered an essential part of the CR action logits.

This mask corresponds to the input slots and identifies which decision heads represent real conflicting control decisions and which correspond to padding. For example, if the network is configured with $N_{\text{ConfDec}} = 3$ slots but only 2 conflicting decisions are present in the current inference step, the validity mask would be structured as:

$$\mathbf{m} = [1, 0, 1], \tag{6.4}$$

where 1 indicates a valid decision slot and 0 indicates padding.

During post-processing, this mask is applied to the output logits; invalid slots are masked with negative infinity ($-\infty$) to ensure that their resulting probability after the Softmax operation is

effectively zero. This mechanism ensures that ACCoRD ANN's policy is derived solely from valid control decisions and prevents the network from training on padding noise.

Critic value

As ACCoRD employs an actor-critic RL architecture (specifically PPO-Clip), the ANN includes a secondary output known as the critic value. In parallel with the decision heads, the critic estimates the state value function $V(s)$, representing the expected reward from the current state s.

ACCoRD formulates the conflict resolution task as a contextual bandit problem, which serves as an intermediate between the multi-armed bandit and full reinforcement learning [70]. In this ML approach, the agent observes a specific state s and selects an action to maximize the immediate reward. Unlike full RL, this formulation assumes a single-step optimization horizon, effectively disregarding how the current action influences future state transitions or delayed rewards.

The motivation for adopting this formulation stems from the specific operational nature of conflict resolution in O-RAN. Unlike continuous control tasks where actions drive the system through a connected trajectory of states, conflict episodes handled by ACCoRD are often decoupled time-wise and space-wise: a CR action applied to a specific set of conflicting decisions (e.g., at a cell on the network edge) has negligible causal influence on the state of a different conflict occurring hours later or in a geographically distant part of the network. Consequently, maximizing the immediate reward for the current specific context is considered as the optimal objective.

Therefore, the critic learns to predict the expected immediate reward for a given network state s, rather than a discounted sum of future returns. The value function is defined as:

$$V(s) = \mathbb{E}\left[r_t \mid s_t = s, \pi_\theta\right], \tag{6.5}$$

where r_t is the reward received at time step t following the action taken by policy π_θ.

The critic head processes a concatenated vector of the global environment features and the pooled context of the conflict slots. This value output serves as the baseline for calculating the advantage function, defined as the difference between the action-value and the state-value functions ($A(s,a) = Q(s,a) - V(s)$) [157]. By subtracting this state-dependent baseline from the observed reward, the variance of the policy gradient updates is significantly reduced, stabilizing the learning process. While this value output does not directly influence the immediate action selection during inference, it is crucial for the stability of the training.

6.2.3 ANN structure

The ANN implemented in ACCoRD is a feed-forward actor-critic network. In such architecture, the actor is responsible for learning the policy $\pi_\theta(a|s)$ (selecting the best action given the state), while the critic learns the state value function $V(s)$ (estimating the expected return from the current state) used to compute advantages. The internal structure of the ANN with its inputs and outputs is shown in Figure 6.3.

The ANN is composed of three distinct functional blocks, as depicted in Figure 6.3:

- Control decision feature encoder - normalized input features for each conflicting decision are encoded via a shared two-layer MLP with Rectified Linear Unit (ReLU) activation. This module projects the low-dimensional scalar parameter values into a dense latent representation suitable for processing by the decision heads.

- Control decision heads - each head combines the encoded decision features with the parameter type embedding, global cell and user features, and a mean-pooled summary of all other

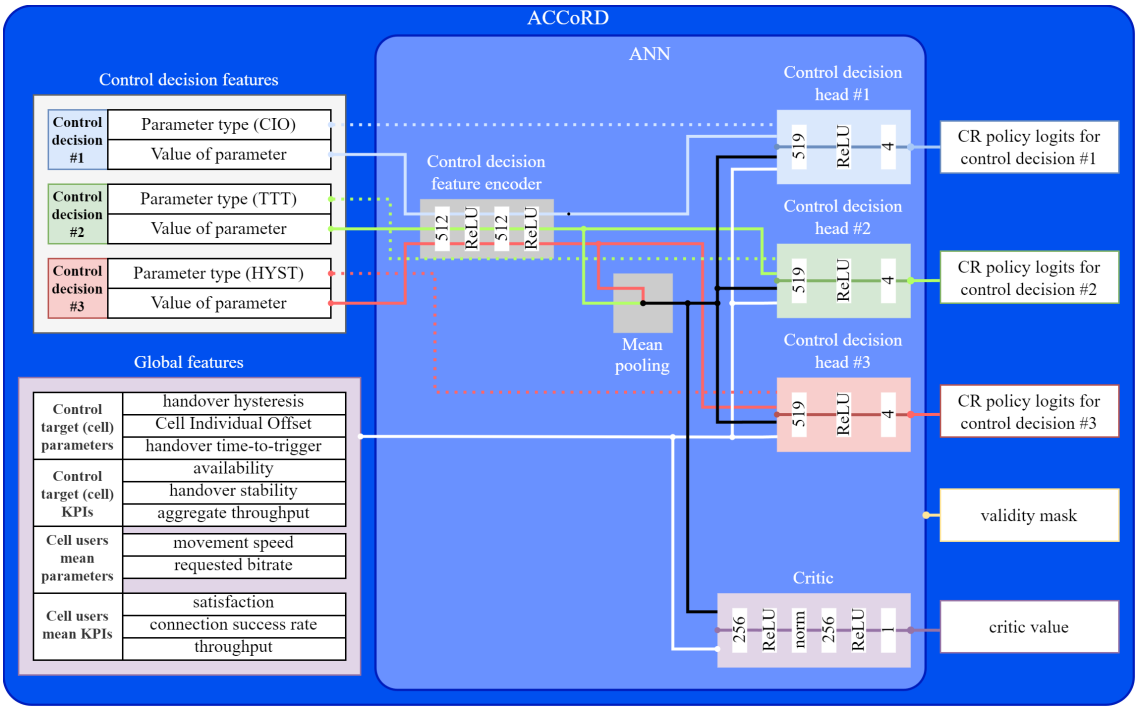

FIGURE 6.3: Structure of the ANN implemented as part of ACCoRD

Source: own elaboration

encoded decisions in the conflict. This context-aware representation allows the head to output CR action preferences specific to its assigned conflicting control decision.

- Critic - a three-layer MLP with ReLU activation and layer normalization; based on the mean-pooled summary of encoded control decision features and the global features, it outputs a single scalar state-value estimate used to compute advantages during training.

The structure is dimensioned to handle up to N_{ConfDec} conflicting control decisions simultaneously. In this implementation, there is only a single input feature describing each control decision: the normalized value of the relevant control parameter. Features for absent decisions are padded with zero values and the `NONE` embedding, and are subsequently excluded from the policy update via the validity mask mechanism described in Section 6.2.2. The encoded control decision features are combined with parameter type embedding, global features, and mean pooling of the other encoded control decision features, and used as input to the relevant control decision head. The mean-pooled encoded control decision features are also combined with global features and fed into the critic element, which produces a critic value used for training of the ANN.

6.2.4 Training process

Training of the ANN is performed using the PPO-Clip RL algorithm [158]. PPO is a policy gradient method designed to balance implementation simplicity with sample efficiency and reliable convergence.

The "Clip" variant specifically addresses the issue of training stability by utilizing a clipped surrogate objective function L^{CLIP}, defined as:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right], \tag{6.6}$$

where $r_t(\theta)$ is the probability ratio between the new and old policies, $\hat{A}_t$ is the estimated advantage function, and ϵ is a hyperparameter that limits ("clips") the magnitude of policy updates. This clipping prevents the agent from making drastically large changes that could lead to performance collapse [158].

Upon detection of a conflict, the environment state observations are made. The collected data are then normalized and fed into the ANN for inference. ACCoRD interprets the ANN's outputs and provides the altered control decisions to the CR Agent for execution, effectively applying these actions to the controlled environment, which is the O-RAN network. The data flow for ACCoRD with focus on the RL training process is illustrated in Figure 6.4. The diagram is showing the state-action-reward-state view of ACCoRD and the environment (O-RAN network), similarly to how RL is presented in Figure 2.9. However, the detail here extends into the internal communication of ACCoRD. The information exchanges were color-coded as follows: state-related data is described on purple background, reward-related data on green background, and action-related data on red background.

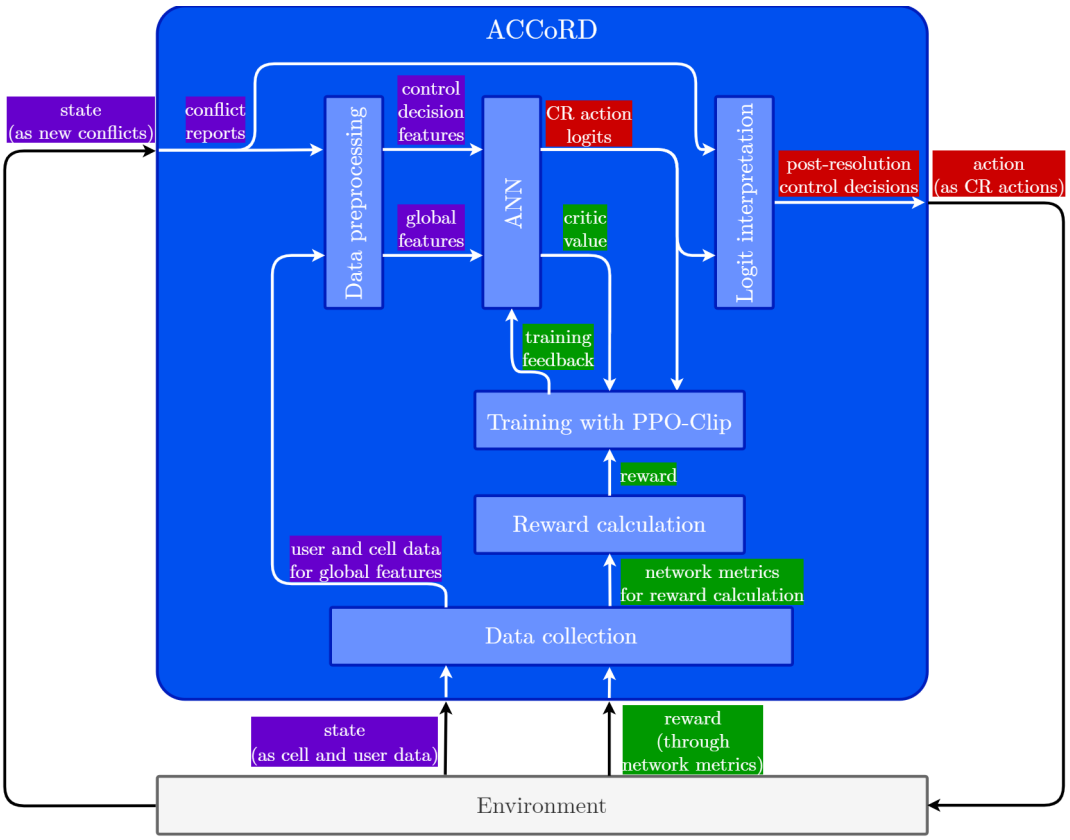

FIGURE 6.4: Data flow for ACCoRD's RL training

Source: own elaboration

Once the CR actions are implemented, ACCoRD schedules a post-resolution observation of the network state. During the post-resolution observation, it calculates the reward value for the ANN. The details of the reward calculation are described in Section 6.2.5. State information related to each inference of the ANN is stored in an experience buffer. Invalid slots (i.e., slots for decisions with control parameters not present in the conflict) are masked and ignored at each step of training process.

Experience is collected in 400 s intervals throughout each training run. At the end of each interval, the observations are split into 64-element minibatches and used to update ANN's policy over 3 PPO epochs.

Reward values are normalized using a running normalizer to mitigate fluctuating changes in reward distribution. Furthermore, to ensure the training process remains stable, ACCoRD monitors the Kullback-Leibler (KL) divergence between the policy before the update ($\pi_{\theta_{old}}$) and the current policy (π_θ). High divergence indicates "policy drift," where ACCoRD's behavior changes too rapidly, potentially leading to loss of gained experience. The approximate KL divergence is calculated as:

$$D_{KL} \approx \hat{\mathbb{E}}_t \left[\log \pi_{\theta_{old}}(a_t|s_t) - \log \pi_\theta(a_t|s_t) \right]. \tag{6.7}$$

This metric drives an adaptive learning rate scheduler. If D_{KL} exceeds $1.5 \times$ target_KL, the learning rate is reduced to stop the policy drift by stabilizing the updates. Conversely, if D_{KL} is significantly lower than the target, the learning rate is increased to accelerate convergence.

Consecutive training steps with significantly low or high KL divergence lead to an adequate adjustment of learning rate (in the range from 10^{-6} to $5 \cdot 10^{-4}$) and PPO clip (in the range from 0.08 to 0.12). In case of good stability of subsequent steps of the training process, an additional PPO epoch is added to exploit the low KL divergence and, in result, extract a larger policy update from the same experience data. On the other hand, observed values of running average of KL divergence exceeding 150% of KL target lead to early termination of the epoch.

Each batch training includes a 12-step pretraining of the critic with Huber loss. This specific number of steps was determined empirically to provide a stable initial estimate of the value baseline without overfitting to the current batch, ensuring that the advantage calculation for the subsequent policy update is grounded in a reasonably accurate critic. The Huber loss is defined as:

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \leq \delta, \\ \delta(|y - f(x)| - \frac{1}{2}\delta) & \text{otherwise,} \end{cases} \tag{6.8}$$

where δ is a hyperparameter defining the transition between quadratic and linear loss.

Critic gradients are clipped to avoid large updates. One-step advantages are computed based on normalized rewards and a snapshot of critic values gathered at beginning of the training process. The calculated advantage values are normalized to mean 0.0 and standard deviation 1.0 across the entire batch to reduce variance and further stabilize policy update steps.

Once the advantages are computed, the CR Agent executes PPO epochs to update the ANN's policy. Each PPO epoch operates on minibatches in a random order to avoid bias from fixed positions of data during learning. A forward pass is performed to obtain current policy logits and critic value predictions; invalid slots (i.e., empty slots not related to any control decision in the conflict) are masked to not skew the policy updates. Huber loss of the current policy is computed based on clipped value loss between old and new policy. An entropy factor is considered during loss calculation to encourage exploration of the ANN in the first phase of training. Similarly as during training of the critic, gradient norms are clipped to limit large ANN updates in single steps. Finally, the epoch concludes with a training step of the ANN performed using an AdamW optimizer for both actor and critic.

The values of hyperparameters were chosen empirically by running test runs and fine-tuning the relevant values to achieve a stable and relatively fast progress during the learning process. The relevant hyperparameters are shown in Table 6.4.

6.2.5 Reward calculation

To calculate the reward value, ACCoRD captures metrics that describe the performance of the entire network as the reward signal reflects the impact of CR decisions on global network performance. ACCoRD employs a differential measurement strategy to isolate the specific effect of

Table 6.4: ANN training hyperparameters

Name	Value	Description
ppo_epochs	3	PPO epochs per update (on the same on-policy batch).
max_extra_epochs	1	Extra PPO epochs for when stability criteria are met.
ppo_minibatch_size	64	Minibatch size for PPO inner loop.
ppo_clip	0.10	Starting PPO clipping parameter.
ppo_min_clip	0.08	Lower bound for PPO clipping parameter.
ppo_max_clip	0.12	Upper bound for PPO clipping parameter.
target_kl	0.015	Target mean KL per update (used for early stop).
actor_lr	1×10^{-5}	Starting learning rate for actor optimizer.
actor_min_lr	1×10^{-6}	Lower bound for actor LR.
actor_max_lr	1×10^{-3}	Upper bound for actor LR.
critic_lr	1×10^{-4}	Starting learning rate for critic optimizer.
critic_min_lr	1×10^{-6}	Lower bound for critic LR.
critic_max_lr	2×10^{-4}	Upper bound for critic LR.
critic_n_pretrain_steps	12	Critic initial training steps per update.
critic_value_clip	0.2	Critic value clipping coefficient.
Huber_delta	1.0	Huber loss δ for actor and critic loss.
max_grad_norm_clip	0.5	Grad-norm clipping for actor and critic.
entropy_coeff	0.007	Entropy bonus coefficient.
value_coeff	1.0	Value loss weight in total objective.
reward_normalization_alpha	1×10^{-3}	EMA factor for rewards normalization.
reward_normalization_eps	1×10^{-8}	Numerical epsilon for rewards normalization.

a resolution action. Metrics are observed within a rolling measurement window of length T_{meas} at two distinct time points for each detected conflict:

1. Baseline snapshot ($t = 0$): taken at the moment of conflict detection and immediate execution of CR actions. This snapshot aggregates metrics over the interval $[-T_{\text{meas}}, 0]$, capturing the state of the network performance immediately prior to the resolution.

2. Result snapshot ($t = T_{\text{reward}}$): taken after a specific delay T_{reward} following the application of CR actions. This snapshot aggregates metrics over the interval $[T_{\text{reward}} - T_{\text{meas}}, T_{\text{reward}}]$. In the standard configuration where $T_{\text{reward}} = T_{\text{meas}}$, this snapshot effectively captures the network performance over the period $[0, T_{\text{meas}}]$, representing the immediate effects of the resolution actions.

ACCoRD then computes the difference between these two snapshots to derive the reward. To ensure that the observation captures only the immediate effects of the resolution actions, without interference from subsequent control decisions issued by xApps in the next cycle, the observation delay and measurement window are tightly synchronized with the xApp operation. The following metrics are captured for reward calculation during measurement window:

- ping-pong handover count (C_{PP}) - total count of ping-pong handover events in the network; acts as an indicator of user mobility instability.

- RLF count (C_{RLF}) - total count of RLF events in the network; acts as an indicator of user mobility failure.

- CB count (C_{CB}) - total count of CB events in the network; acts as an indicator of network capacity/availability issues.

To ensure training stability, all raw metrics are normalized into the range $[0.0, 1.0]$. For that purpose, ACCoRD tracks the minimum ($C^{\min}$) and maximum ($C^{\max}$) values observed during runtime. The normalized value $\hat{C}$ for a given metric is computed as shown in (6.9).

$$\hat{C} = \frac{C - C^{\min}}{C^{\max} - C^{\min}}. \tag{6.9}$$

The final reward R_{network} is derived from the weighted sum of the changes in these normalized metrics ($\Delta\hat{C}$), as defined in (6.10).

$$R_{\text{network}} = -\frac{w_{\text{PP}}^{\text{rew}}\Delta\hat{C}_{\text{PP}} + w_{\text{RLF}}^{\text{rew}}\Delta\hat{C}_{\text{RLF}} + w_{\text{CB}}^{\text{rew}}\Delta\hat{C}_{\text{CB}}}{w_{\text{PP}}^{\text{rew}} + w_{\text{RLF}}^{\text{rew}} + w_{\text{CB}}^{\text{rew}}}, \tag{6.10}$$

where $w_{\text{PP}}^{\text{rew}}$ denotes the weight applied to the $\hat{C}_{\text{PP}}$ normalized metric according to the current policy of the RIC. Analogously, $w_{\text{RLF}}^{\text{rew}}$ and $w_{\text{CB}}^{\text{rew}}$ weight the $\hat{C}_{\text{RLF}}$ and $\hat{C}_{\text{CB}}$ normalized metrics, respectively. These weights reflect the operator's intent and provide an interface for the operator to tweak operation of ACCoRD to align with the intent. For example, setting a relatively high weight for RLF prioritizes connection reliability over load balancing. The negative sign ensures that increases in adverse events (positive $\Delta\hat{C}$) result in lower rewards, incentivizing the agent to minimize these network faults.

6.2.6 Parametrization of ACCoRD

The operation of ACCoRD is configured by a set of parameters that define the dimensions of the ANN, the granularity of the action space, and the temporal aspects of the interaction with the O-RAN environment. These parameters are categorized into structural parameters, which define the architecture of the ANN, and operational parameters, which dictate the training and inference cycles.

Structural parameters

The primary structural constraints of the ANN are determined by the maximum complexity of conflicts the system is designed to handle and the resolution available to the agent.

- Maximum number of conflicting control decisions (N_{ConfDec}): defines the number of input slots and corresponding output heads in the ANN. For the evaluation scenario involving MRO and MLB conflicts, this is set to $N_{\text{ConfDec}} = 3$, as the largest expected conflict involves a clash of CIO, TTT and hysteresis reconfiguration.
- Number of CR actions (N_{A}): defines the number of neurons in the final layer of each decision head. As detailed in the Section 6.2.2, $N_{\text{A}} = 4$, corresponding to the following CR actions: `NO_MODIFICATION`, `REJECTION_WITH_COOLDOWN`, `INCREASE_1`, and `DECREASE_1`.

Operational parameters

The interaction between ACCoRD and the network environment relies on specific timing intervals to ensure stable learning and accurate reward estimation.

- Measurement window (T_{meas}) and reward delay (T_{reward}): these parameters define the metric aggregation period and the time offset for the post-resolution snapshot, respectively. To ensure that ACCoRD captures only the immediate effects of resolution actions, without interference from subsequent control decisions in the next xApp cycle, both parameters must be synchronized with the smallest processing interval of the active xApps. For this evaluation, they are set to $T_{\text{meas}} = T_{\text{reward}} = 1.0\,\text{s}$, as both xApp operate in 1.0 s intervals.
- Cooldown duration (t_{CR}): the duration for which a source xApp is blocked from issuing further updates if the `REJECTION_WITH_COOLDOWN` action is selected. This is set to $t_{\text{CR}} = 10\,\text{s}$ in alignment with the configuration of the prioritization with cooldown method depicted in Section 5.4.7.

- Training rollout interval (t_{training}): the duration of experience collection before a PPO training update is triggered. ACCoRD collects trajectories for 400 s before performing a policy update based on collected experience batch.

Reward weights

The reward function, defined in Eq. 6.10, is parametrized by weights that determine the prioritization of different network faults. In the balanced policy configuration used for evaluation, these weights are set equally: $w_{\text{PP}}^{\text{rew}} = 1.0$, $w_{\text{RLF}}^{\text{rew}} = 1.0$, and $w_{\text{CB}}^{\text{rew}} = 1.0$.

6.2.7 Evaluation of the proposed AI/ML-based conflict resolution method

The evaluation of ACCoRD is performed using the same simulation environment and scenario as the baseline rule-based methods described in Sections 5.4.4, configured with the structural and operational parameters defined in Section 6.2.6. The scenario involves a cluster of 7 cells serving a varying number of UEs (Small, Medium, and Large deployment configurations). The network is optimized simultaneously by MRO and MLB xApps, which generate indirect parameter conflicts regarding CIO, TTT, and handover hysteresis.

ACCoRD is deployed as the decision-making actor within the CR Agent. Its performance is judged against a set of baselines. Method *prioritization with cooldown* is used as the baseline, since its variant prioritizing MRO was the best-performer in the results presented in Section 5.4.7, as well as a baseline case with no ConMit measures.

Experimental procedure

Unlike static rule-based methods, ACCoRD requires a learning period to adapt its policy to the network dynamics. To address this, the evaluation procedure for each simulation run follows a two-phase structure applied to identical network scenarios:

1. Training phase (10,000 s): the simulation is initialized with a specific random seed determining UE placement and traffic patterns. It runs for an extended duration to allow the ANN to explore the state-action space. During this phase, actions are sampled stochastically from the policy distribution π_θ to encourage exploration. ACCoRD collects experience and updates ANN's weights via the PPO-Clip algorithm every 400 s (t_{training}).

2. Evaluation phase (1,000 s): upon completion of training, the policy is frozen (weights are no longer updated). The simulation is reset to the exact same initial state as the training phase (using the identical random seed). This phase runs for 1,000 s, during which ACCoRD exploits the learned policy by deterministically selecting the action with the highest probability. The metrics presented in this section are derived exclusively from this separate evaluation run.

To ensure stability, the first 200 s of the evaluation phase are treated as a warm-up period, and measurements from this interval are discarded. This process is repeated for 36 random UE placement seeds per deployment configuration to ensure statistical robustness.

Results and analysis

Table 6.5 summarizes the outcomes, showing the relative penalty metric for each CR method averaged over 36 simulation runs per each of the three considered UE deployment configurations, with each run using a randomized initial placement of UEs. The column "Delta to best" contains

Table 6.5: Evaluation results based on 36 simulation runs within each UE deployment scenario.

(Lower penalty and delta are better)

Rank	CR method	Average penalty	Average relative penalty (%)	Delta to best (pp)
	Small UE deployment configuration			
1	Prioritization of MRO with cooldown	292.13	102.5	0.0
2	**ACCoRD**	297.73	104.1	1.6
3	ConMit disabled	302.97	106.3	3.8
4	Prioritization of MLB with cooldown	330.43	116.0	13.5
	Medium UE deployment configuration			
1	**ACCoRD**	402.84	101.4	0.0
2	Prioritization of MRO with cooldown	412.82	104.2	2.8
3	ConMit disabled	424.45	107.1	5.7
4	Prioritization of MLB with cooldown	448.79	113.2	11.8
	Large UE deployment configuration			
1	**ACCoRD**	549.87	101.6	0.0
2	Prioritization of MRO with cooldown	568.56	105.6	4.0
3	ConMit disabled	583.45	108.3	6.7
4	Prioritization of MLB with cooldown	602.99	111.9	10.3

the difference in average relative penalty from the best method in given UE deployment scenario, in percentage points (pp).

In the small UE deployment configuration, the simpler rule-based method (Prioritization of MRO with cooldown) performs best, achieving a relative penalty score of 102.5%. ACCoRD follows closely at 104.1% ($\delta = 1.6$ pp). This suggests that in low-complexity environments, the overhead of learning a policy may slightly outweigh the benefits compared to a well-tuned static rule.

However, the robustness of ACCoRD starts showing as network complexity increases. In the medium UE deployment scenario, ACCoRD achieves a score of 101.4% and outperforms Prioritization of MRO with cooldown by 2.8 *pp.* The other two methods trail behind in the same order as in the small deployment configuration: ConMit disabled sits at 107.1% ($\delta = 5.7$ *pp.*) and Prioritization of MLB with cooldown at 113.2% ($\delta = 11.8$ *pp.*).

The gap widens further in the large UE deployment configuration, where ACCoRD scores 101.6%, with gap to the next best method Prioritization of MRO with cooldown (sitting at 105.6%) widened to 4.0 *pp.* Replicating the further order from small and medium UE deployments, ConMit disabled and Prioritization of MLB with cooldown scored 108.3% ($\delta = 6.7$ *pp.*) and 111.9% ($\delta = 10.3$ *pp.*), respectively.

The results demonstrate that the efficiency of ACCoRD scales favorably with network load and conflict frequency. While simple prioritization strategies can offer improvements over an unmanaged network (as seen with Prioritization of MRO with cooldown), they are rigid; the incorrect choice of priority (e.g., prioritization of MLB) can significantly degrade performance (111.9% − 116.0%). In contrast, ACCoRD dynamically adapts to the challenging conditions of high-traffic networks to consistently minimize penalties.

Simple prioritization with cooldown can improve network performance in comparison to lack of ConMit measures (as in case of Prioritization of MRO with cooldown). However, the choice of prioritized xApp matters: Prioritization of MLB with cooldown consistently performs worst by a wide margin and in this case the prioritization with cooldown method significantly deteriorates network performance.

The conclusion regarding the optimal deployment of CR methods in CMF is that for low traffic scenarios a simple rule-based prioritization method can be employed to achieve good results while saving on computation resources. In medium or high traffic cases, ACCoRD shall be deployed due to its advantage in performing in challenging conditions. Alternatively, ACCoRD can be deployed at all times, including low traffic cases, to gather more data in such deployment scenarios and refine its policy over time.

Table 6.6: Statistics for simulation run #1 in medium UE deployment configuration ($t \geq 200\,\mathrm{s}$)

Algorithm	C_{HO}	C_{PP}	C_{RLF}	C_{CB}	Penalty	Rel. Penalty
ACCoRD	**8378**	**3856**	507	**59**	**419.20**	**100.0**
Prio. MRO	8859	4362	**448**	85	431.30	102.9
ConMit disabled	8634	4346	506	77	450.50	107.5
Prio. MLB	8913	4618	516	74	466.90	111.4

6.2.8 Case study: conflict resolution policy derived by ACCoRD

We examine an example policy derived by ACCoRD in simulation run #1 with medium UE deployment configuration. Simulation results with metrics relevant to calculation of the penalty metric are shown in Table 6.6. In this simulation run, ACCoRD ranks best with the lowest penalty score at 419.20.

During 800 seconds of this simulation run, we have observed 926 control conflicts between MLB and MRO. The best-performing policy derived by ACCoRD follows these three rules to solve them:

- conflicting decisions regarding CIO are always rejected, regardless of other parameters. This rule effectively stabilizes cell handover offsets during conflicts.
- conflicting decisions regarding TTT and hysteresis are either rejected or increased by a single step; rejection of these decisions pause the handover parameters to stabilize the network situation, while increasing the parameters lead to increased inertia of the handovers (for TTT due to longer time required for handover; for hysteresis due to larger needed signal delta for triggering handover).
- overall, in the majority of conflicts (nearly 64%), all conflicting control decisions are rejected; ACCoRD clearly seems to prioritize stabilization of handover parameters until decision conflict passes over time.

The ACCoRD-derived policy, in this case, acts mostly as a stabilizer of the handover dynamics.

Linking these rules to the apparent performance of the methods presented in Section 6.2.7, ACCoRD achieves the lowest counts of regular and ping-pong handovers. Moreover, despite the decreased handover frequency, ACCoRD avoids CBs by reaching a significantly lower number of these events. Simultaneously, the RLF count is kept at a competitive level similar to ConMit disabled case.

Chapter 7

Hardware evaluation of Conflict Mitigation Framework

As demonstrated in the previous chapters, the current landscape of the O-RAN Alliance specifications and related research primarily evaluates ConMit solutions through computer simulations or RF channel emulation (like in Colosseum [81, 82]). While simulation-based evaluations, such as the one presented for ACCoRD in Chapter 6, provide insight into algorithmic efficiency of the proposed solutions, they cannot fully capture the stochastic nature of real-world RF environments.

To this end, this chapter presents the experimental evaluation of the proposed CMF in a real-world O-RAN deployment. To the best of the author's knowledge, this constitutes the first evaluation of a ConMit method on an O-RAN-compliant testbed utilizing OTA RF transmission between the RAN and commercial UEs. This evaluation builds upon the co-authors' previous work on hardware-based ConMit prototyping [148] and an initial implementation of time-based conflict mitigation (i.e., first-come, first-served) with single UE [159]. By expanding these efforts to a dynamic environment with multiple UEs, the work presented in [4] provides a comprehensive performance assessment of the CMF implemented with two custom conflicting xApps. The experimental validation is conducted using the Virginia Tech Commonwealth Cyber Initiative (CCI) xG testbed [160]. The primary contributions of this hardware evaluation are as follows:

1. Deployment of a real-world O-RAN environment with direct control conflicts - an O-RAN network is deployed using open-source components, including srsRAN and Open5GS, hosting two purpose-built xApps designed to generate direct control conflicts over slice-level resource allocation, and handling communication with UEs through OTA transmission.

2. CMF deployed in the environment - the CMF described in Chapter 5 is implemented within the hardware-based Near-RT RIC; it includes both the CD function via the DCD procedure and the CR function via prioritization of xApp #1, enabling complete ConMit operation for the considered case of direct conflict.

3. Performance assessment of the deployed solution - the performance evaluation of CMF is done through multiple test-runs by comparing a performance metric, i.e., UE-level Downlink (DL) throughput, of the hardware-based network with and without the ConMit measures in place.

7.1 System model

The objective of this experiment is to validate CMF's ability to detect and resolve direct conflicts in a system architecture compliant with O-RAN specifications, utilizing COTS hardware and open-source software stacks. The system diagram of the deployed testbed is illustrated in Figure 7.1.

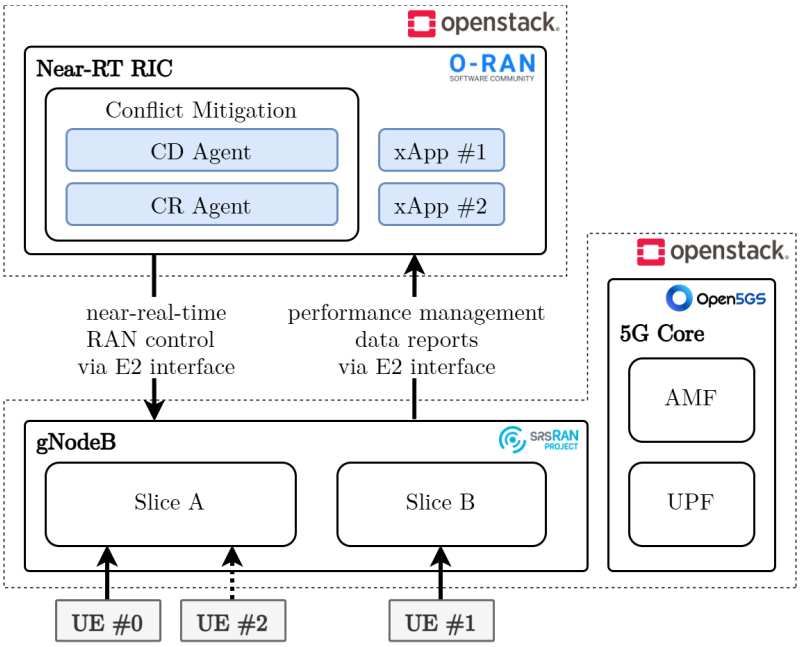

FIGURE 7.1: End-to-end system diagram of the testbed setup

Source: own elaboration based on [4]

The deployed system comprises a single gNodeB operating on the 3600 MHz frequency. This single cell supports two distinct network slices, named A and B, which both handle eMBB traffic. These slices are allocated a portion of the cell's radio resources. Slice A aims to provide services to prioritized UEs, while slice B handles traffic from regular non-prioritized UEs. Up to three UEs connect to the deployed cell in the considered system model. Within this user group, UEs #0 and #2 are prioritized and connect to slice A, while UE #1 is lower-priority and connects to slice B.

The deployed Near-RT RIC hosts the two custom Python-based xApps designed to optimize slice-level PRB allocations utilizing Action 6, Style 2 of the Control service defined in E2 Service Model-RAN Control (E2SM-RC) [58]. Although both xApps operate on the same control parameters, they pursue distinct optimization targets, leading to a direct control conflict between them. The control logic for each was designed to demonstrate a practical, effective steering approach toward these specific outcomes, rather than to represent a theoretically optimal implementation.

The primary objective of xApp #1 is to prioritize traffic for UEs connected to slice A. It achieves this by allocating a proportionally larger share of the available resources to this slice. The allocation logic balances slice priority (based on user count) with a baseline fairness factor. Let the total number of available PRBs be P, the total number of served UEs be U, and the total number of slices be S. For the prioritized slice A serving U_A users, xApp #1 calculates the priority PRB ratio (r_A) and the final allocation ($p_{1,A}$) as shown in (7.1).

$$r_A = \frac{U_A}{U}, \qquad p_{1,A} = P \cdot \frac{r_A + 1/S}{2}. \tag{7.1}$$

For all other non-prioritized slices, the remaining resources are distributed equally. The non-priority ratio (r_s) and the resulting allocation ($p_{1,s}$) are calculated according to (7.2). This equation applies only if there are multiple slices ($S > 1$)

$$r_s = \frac{1 - r_A}{S - 1}, \qquad p_{1,s} = P \cdot \frac{r_s + 1/S}{2}. \tag{7.2}$$

In this design, the final allocation is the average of the specific demand ratio (r_A or r_s) and the equal distribution factor ($1/S$). This mechanism ensures the logic effectively steers resources toward slice A while maintaining overall fairness by preventing complete resource starvation of other slices.

In contrast, xApp #2 aims to enforce an even distribution of resources across all network slices within the gNodeB, regardless of user load. The PRB allocation p_2 for any given slice is calculated according to (7.3).

$$p_2 = P/S. \tag{7.3}$$

Both xApps are active concurrently within the network. Their respective logic is triggered periodically every 10 seconds, after which they compose and transmit E2 Control messages to enforce the calculated allocations. More detailed view of their logic and examples of calculation of the slice-level resource allocations for both implemented xApps are shown in Section 7.4.

To manage the interactions of the two deployed applications, a ConMit component is implemented within the Near-RT RIC. It adheres to the architecture envisioned in the CMF [2], as it implements the CD Agent and CR Agent. The CD Agent employs the DCD procedure to detect conflicts between xApps #1 and #2, and the CR Agent is configured with prioritization method of conflict resolution, with the higher priority set for xApp #1. The conflict scenario is described in more detail in Section 7.3.

The system also includes a 5G Standalone core component responsible for providing simple network backbone functionality. The deployed 5G Core hosts Access and Mobility Management Function (AMF) and User Plane Function (UPF). The former handles connectivity and mobility management, while the latter carries user data packets between the gNodeB and the backbone. This setup allows for basic RAN operation and enables the UEs to connect to the testbed as to a commercial O-RAN network.

7.2 Hardware implementation of the system

The system setup presented in Figure 7.1 was implemented in a dedicated O-RAN testbed, the CCI xG Testbed [160], set up in the North American OTIC in Washington DC/Arlington VA. The architecture of the configured network used as a testbed consists of components compliant with O-RAN specifications.

The main RAN component, the gNodeB, runs on a dedicated OpenStack Virtual Machine (VM). For the RF frontend, a Universal Software Radio Peripheral (USRP) x310 configured with 20 MHz bandwidth is utilized. Located in the Radio Ceiling Grid (the indoor component of the CCI xG Testbed), this unit features an antenna configuration with 2 transmitters and 2 receivers, enabling 2x2 MIMO operation. On the software side, the gNodeB functionality is provided by the Software Radio Systems RAN (srsRAN) 5G stack [161] deployed within the VM.

The UEs are commercial off-the-shelf hardware. Specifically, 3 Samsung mobile phones equipped with programmable SIM cards were used to generate traffic and connect to the network.

The Near-RT RIC is also hosted on an OpenStack VM, albeit separate from the one running the gNodeB. Regarding the software platform, the O-RAN SC Near-RT RIC [77] is deployed as a Docker container [162], equipped with the necessary interfaces to communicate with the RAN node.

The xApps are similarly on-boarded as Docker containers within this platform, ensuring compatibility with the RAN environment and its service models.

Finally, the mobile core network deployment is co-located on the same OpenStack VM as the gNodeB. It is implemented using the Open5GS Core software stack [163] to complete the network architecture.

A photo of the experimental equipment in the testbed environment is provided in Figure 7.2.

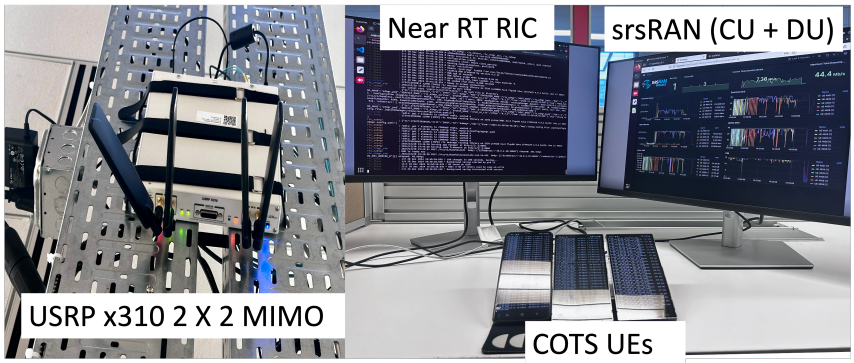

Figure 7.2: Experimental setup in the CCI xG Testbed

Source: [4]

7.3 Conflict scenario and mitigation approach

The control conflict between xApps in the considered scenario is a direct conflict between xApp #1 and xApp #2, originating from both xApps modifying the slice level PRB allocations with different optimization goals. A detailed view of the operation of the two considered xApps is provided in Section 7.4.

As mentioned in Section 7.1, the Near-RT RIC in the scenario implements CMF with DCD and CR through basic prioritization. As such, the conflict between xApps #1 and #2 is detected according to the proposed DCD procedure described in Section 5.4.3: the CR Agent tracks all control decisions of both xApps and validates if any of the decisions are made against the same control target. If such conflicting control decisions are found, then the information about the conflict is passed into the CR Agent for resolution. This procedure in context of the considered scenario is shown in Figure 7.3.

The CR approach is tailored to the specific roles of the xApps deployed in the system. xApp #1 is responsible for optimizing network performance for high-priority traffic in slice A, making it the preferred xApp in cases of control conflicts. When both xApp #1 and xApp #2 issue E2 control messages that are active simultaneously and target the same control target (i.e., slice), the CD Agent identifies the conflict as direct. The CD Agent forwards an adequate conflict report to the CR Agent, which, in turn, consistently resolves the conflict by prioritizing xApp #1 and rejecting the control decision from xApp #2. As a result, only the control action from xApp #1 is applied to the network, ensuring that the high priority traffic optimization is not affected by competing xApp decisions.

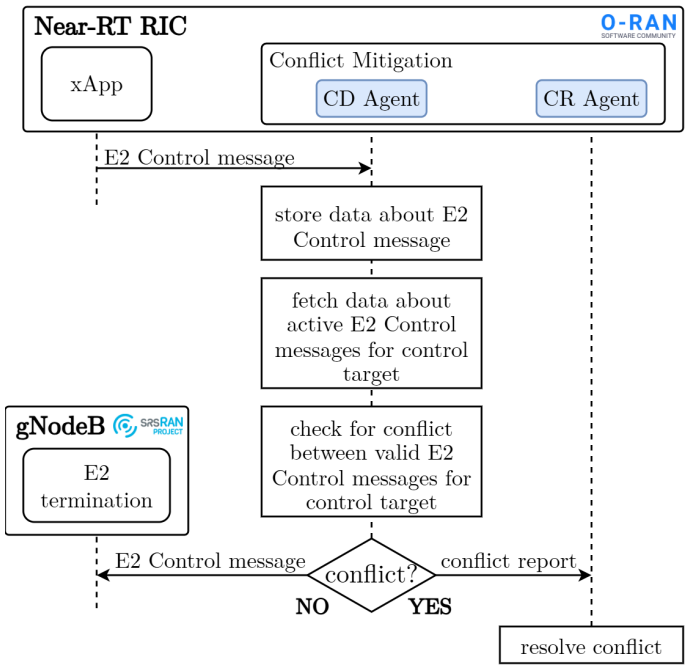

FIGURE 7.3: Conflict mitigation procedure in the implemented testbed

Source: own elaboration based on [4]

7.4 Operation of custom-developed xApps for slice-level PRB allocation

To complement the theoretical description of the xApp control logic presented in Section 7.1, this section details the algorithmic implementation of the two custom xApps. The following pseudocode listings and numerical examples illustrate the exact procedural steps executed by the applications during each control interval.

For the purpose of the examples provided below, the following network state is assumed, corresponding to the second phase of the conflict scenario described in Section 7.3 (i.e., when all three UEs connect to the network):

- Total available PRBs (P): 273 (typical for 100 MHz bandwidth with 30 kHz SCS).
- Total connected UEs (U): 3.
- UEs in slice A (U_A): 2.
- UEs in slice B (U_B): 1.

7.4.1 Implementation of xApp #1

The logic for xApp #1 is implemented to weigh the slice priority against a baseline fairness factor; slice A is the prioritized one in the considered scenario. Algorithm 1 presents the pseudocode for determining the PRB allocation.

Algorithm 1 Resource allocation logic for xApp #1 (priority-driven)

Require: List of configured slices S, Total available PRBs P
Step 1: Calculate total active users
$U \leftarrow 0$
for each slice $s \in S$ **do**
  $U \leftarrow U + \text{count\_users}(s)$
end for
Step 2: Get users in prioritized slice
$U_A \leftarrow$ count_users(slice A)
Step 3: Calculate baseline average ratio
$r_{\text{avg}} \leftarrow 1/\text{length}(S)$
Step 4: Calculate priority ratios
$r_A \leftarrow U_A/U$
$r_s \leftarrow (1 - r_A)/(\text{length}(S) - 1)$
Step 5: Calculate final allocation per slice
for each slice $s \in S$ **do**
  if s is slice A **then**
    $Alloc_s \leftarrow 0.5 \times P \times (r_A + r_{\text{avg}})$
  else
    $Alloc_s \leftarrow 0.5 \times P \times (r_s + r_{\text{avg}})$
  end if
  Output: Send E2 Control message for slice s with value $\lfloor Alloc_s \rfloor$
end for

Example Calculation for xApp #1:
Based on the assumed network state, xApp #1 executes the following calculations:

1. **UE Counts:** The total user count is calculated as $U = 3$. The count for the prioritized slice A is $U_A = 2$
2. **Ratios:**
   - The baseline average ratio is $r_{\text{avg}} = 1/2 = 0.5$
   - The priority ratio for slice A is $r_A = 2/3 \approx 0.67$
   - The ratio for the non-prioritized slice B is $r_s = (1 - 0.67)/(2 - 1) \approx 0.33$
3. **Allocations:**
   - **Slice A:** $0.5 \times 273 \times (0.67 + 0.5) \approx$ **159** PRBs
   - **Slice B:** $0.5 \times 273 \times (0.33 + 0.5) \approx$ **113** PRBs

7.4.2 Implementation of xApp #2

xApp #2 enforces an even distribution of resources, regardless of slice priorities. Implementation of its logic is straightforward, as detailed in Algorithm 2.

Algorithm 2 Resource allocation logic for xApp #2 (fairness-driven)

Require: List of configured slices S, Total available PRBs P
Step 1: Calculate average ratio per slice
$r_{\text{avg}} \leftarrow 1/\text{length}(S)$
Step 2: Calculate final allocation per slice
for each slice $s \in S$ **do**
  $Alloc_s \leftarrow P \times r_{\text{avg}}$
  Output: Send E2 Control message for slice s with value $\lfloor Alloc_s \rfloor$
end for

Example Calculation for xApp #2:
Regardless of the user distribution ($U_A = 2, U_B = 1$), xApp #2 executes the following:

1. **Ratio:** The average ratio is calculated as $r_{\text{avg}} = 1/2 = 0.5$
2. **Allocations:**
   - **Slice A:** $273 \times 0.5 = \mathbf{136.5} \rightarrow \mathbf{136}$ PRBs (rounded down)
   - **Slice B:** $273 \times 0.5 = 136.5 \rightarrow \mathbf{136}$ PRBs (rounded down)

The discrepancy between the allocation targets of xApp #1 (159/113 PRBs) and xApp #2 (136/136 PRBs) under these conditions results in the direct control conflict managed by the CMF in this experiment. This situation is illustrated in Figure 7.4.

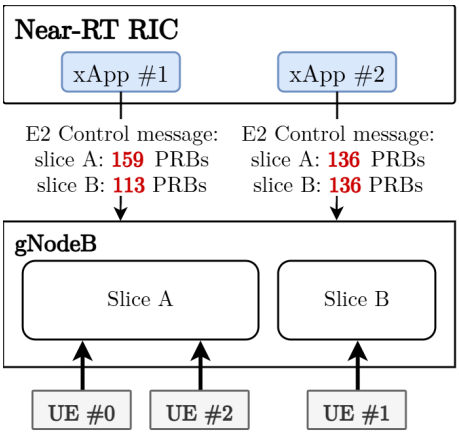

FIGURE 7.4: Example of the conflict scenario in the implemented testbed

Source: own elaboration

7.5 Evaluation scenario

The considered scenario to evaluate the efficiency of the implemented ConMit approach consists of two phases. First, both slices A and B are enabled, and 1 UE is connected to each slice. Then, a third UE connects to slice A. This causes a change in the slice-level PRB allocations computed by both xApps running in the RIC. Due to the logic implemented within the xApps, such circumstance causes a difference in the allocations computed by both xApps, hence leading to a direct control conflict, as described in Section 7.3.

Two cases are considered: without any ConMit measures (*ConMit disabled*) and with ConMit measures in place (*Prioritization of xApp #1*). Ten test runs are carried out for each case. Apart from the ConMit configuration, the testbed configuration remained identical across all test runs. During the experiment, UE-level DL throughput is observed and logged to enable the evaluation of the network performance.

7.6 Evaluation results

The implemented ConMit solution was evaluated through twenty test runs carried out according to the descriptions in Section 7.5. DL throughput for each UE plotted over time during the respective test runs are shown in Figure 7.5.

To ensure reliability of the captured metrics, the analysis excludes the initial transient behaviors associated with network stabilization. Accordingly, the evaluation window is strictly defined to

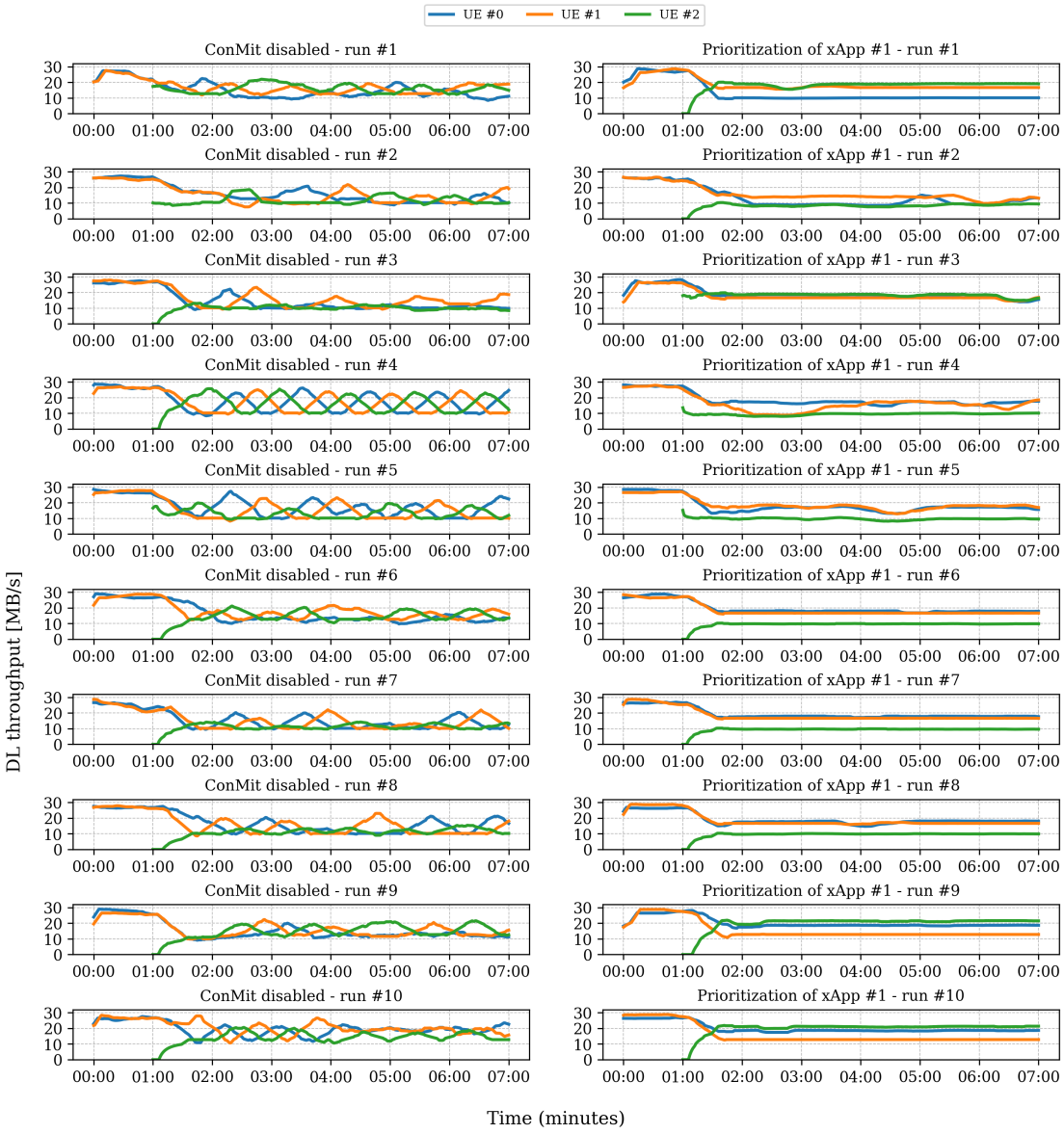

FIGURE 7.5: Downlink throughput measurements captured during each of the test runs

Source: own elaboration based on [4]

start one minute prior to the connection of the third UE to slice A and extends to six minutes thereafter. This approach ensures that the most erratic period of RAN settling—observed at the beginning of the run when only two UEs are connected—is omitted, while still capturing the stable baseline state immediately preceding the conflict event.

Within this valid time window, the average throughput and its standard deviation were calculated for every UE. To identify overall trends, these individual metrics were aggregated to derive run-level statistics, specifically the mean performance across all UEs and the average variability (represented as Standard Deviation (SD)). Finally, the results were consolidated by test category to facilitate a comparison between the *ConMit disabled* and *Prioritization of xApp #1* scenarios. The aggregated results are presented in Table 7.1.

Network operation remained stable across all runs as long as only two UEs were connected, since the logic for both deployed xApps result in the same slice-level PRB allocation settings. However, in scenarios without ConMit measures, significant fluctuations in UE-level DL throughput

Table 7.1: Evaluation of ConMit with CMF and prioritization of xApp #1 in the hardware-based testbed

Category	**Run**	**Downlink throughput** [Mbps]	
		Mean	**Standard deviation**
ConMit disabled	#1	14.769	2.622
	#2	12.567	2.899
	#3	11.972	2.287
	#4	15.984	4.875
	#5	14.143	3.913
	#6	14.603	2.181
	#7	12.673	2.640
	#8	13.007	2.929
	#9	14.473	2.559
	#10	17.443	2.729
	Average	**14.163**	**2.964**
Prioritization of xApp #1	#1	15.117	0.471
	#2	10.731	1.198
	#3	17.545	0.985
	#4	13.481	1.509
	#5	14.400	1.126
	#6	14.773	0.144
	#7	14.734	0.109
	#8	14.726	0.424
	#9	17.579	0.281
	#10	17.445	0.242
	Average	**15.053**	**0.649**

emerged for all UEs following the connection of the third UE to slice A (i.e., after the first minute). For instance, in run "ConMit disabled #4", the DL throughput for UE #1 oscillated between approximately 10 and 24 Mbps. Similar disturbances affected other UEs, signaling unpredictable QoS for all connected devices. This instability stems from unmitigated direct conflicts between xApps #1 and #2 periodically executing contradictory control decisions.

Conversely, in runs with ConMit enabled, the DL throughput for each UE adjusted as the network adapted to new PRB allocations upon the third UE's connection, but subsequently stabilized. Unlike the runs without ConMit, no significant metric fluctuations were observed. This stability results from the ConMit measure prioritizing xApp #1's control decisions over those of xApp #2, enabling the network to reach a steady state.

This contrast in behavior is quantified by the throughput variability, represented by the SD in Table 7.1. The average SD of DL throughput for runs without ConMit measures was 2.964, which is more than 4.5 times higher than the 0.649 SD observed in runs with CMF enabled.

It is worth noting that in specific unmitigated test runs (e.g., Run #10), the mean downlink throughput reached values (17.443 Mbps) comparable to or exceeding the average performance of the mitigated system[cite: 103]. However, this higher mean is driven by extreme oscillations—evident in the high standard deviation of 2.729—rather than sustained performance. While the conflicting xApps occasionally pushed the configuration to high-throughput peaks, the associated variability creates an unpredictable QoE for end-users. In contrast, the CMF approach prioritizes stability, trading off occasional aggressive peaks for a consistent service level.

These findings indicate that the implemented ConMit measures successfully neutralized the negative impacts of the direct conflict between xApps #1 and #2, reducing DL throughput metric variability by 78 %. Consequently, network operation with these measures is significantly more stable and predictable compared to the erratic behavior observed without them. Furthermore, the implemented CD and CR mechanisms demonstrated no negative impact on network resources or total RAN throughput.

Chapter 8

Conclusion

The rapid evolution of RAN towards open, disaggregated, and intelligent architectures, as embodied by O-RAN, presents a paradigm shift in mobile network management. While the introduction of RICs and programmable applications (rApps, xApps, dApps) unlocks unprecedented optimization potential, it simultaneously introduces the critical challenge of control conflicts. This doctoral dissertation set out to prove the central thesis defined in Chapter 1: **conflict mitigation mechanisms can significantly improve the performance, stability, and reliability of O-RAN networks operating with conflicting applications**. The research presented herein confirms this hypothesis through a multi-layered approach involving theoretical analysis, framework design, and rigorous validation via both software simulation and hardware-based experiments.

To address the identified gaps in the state of the art, this work first established a comprehensive categorization of ConMit activities—distinguishing between Preventive ConMit, Conflict Detection and Resolution (CD&R), and Supervision and Adaptation (S&A)—and pinpointed fundamental challenges such as observability and maintenance. Building on this foundation, the author proposed the Conflict Mitigation Framework (CMF), a standardized, logic-agnostic architectural extension for the Near-RT RIC. By introducing dedicated agents for detection and resolution, CMF provides a transparent mechanism to handle conflicts compatible with third-party xApps, avoiding the pitfalls of non-standard, peer-to-peer application coordination.

The validation of the thesis began with the mitigation of direct conflicts, where multiple applications attempt to modify the identical control parameter. The experimental evaluation, conducted on a physical O-RAN-compliant testbed, provided empirical evidence of the thesis in a real-world setting. In the absence of mitigation, the network exhibited severe instability, characterized by extreme oscillations in DL throughput as conflicting control loops fought for dominance. The deployment of CMF with rule-based prioritization successfully neutralized these conflicts, reducing throughput variability by approximately 78 %. This result confirms that for direct conflicts, even relatively simple resolution logic within the proposed framework is sufficient to restore network stability and ensure predictable QoS.

The investigation into indirect conflicts—where applications manipulate different parameters that influence shared KPIs—highlighted the necessity for more advanced resolution strategies. Through extensive simulations comparing rule-based methods against the proposed AI-driven ACCoRD solution, clear performance trends emerged regarding the scalability of conflict mitigation. In scenarios characterized by low traffic volume and low conflict frequency, static rule-based methods proved to be computationally efficient and effective.

However, as the network environment evolved into high-intensity traffic scenarios with frequent and complex conflicts, the limitations of static rules became evident. Under these challenging

conditions, the proposed ACCoRD solution demonstrated superior adaptability. Unlike static policies that may rigidly enforce suboptimal decisions, the RL-based agent dynamically learned to stabilize critical parameters, preventing negative network events impacting the end-users' experience (such as call blockages and drops). Consequently, ACCoRD consistently minimized network penalties across complex deployment scenarios, validating that intelligent, data-driven conflict resolution is essential for maintaining reliability in dense, high-load O-RAN environments.

While this dissertation establishes a robust framework for ConMit, the continuous evolution of the ecosystem suggests further research avenues. Future work should address the latency challenges of mitigating conflicts between neighboring Near-RT RICs and the vertical coordination between RIC layers. Furthermore, it should include evaluation of the proposed CR methods in more complex conflict scenarios, e.g., with more conflicting applications.

In conclusion, the work demonstrates that the thesis is correct: conflict mitigation is not merely an optional feature but a fundamental requirement for the reliable operation of open, intelligent networks. The proposed CMF and associated algorithms offer a viable, standard-compliant path for MNOs to harness the power of O-RAN programmability without compromising network stability.

Own articles

[1] **C. Adamczyk**. Challenges for Conflict Mitigation in O-RAN's RAN Intelligent Controllers. In *2023 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6, 2023. doi:10.23919/SoftCOM58365.2023.10271688.

[2] **C. Adamczyk** and Adrian Kliks. Conflict Mitigation Framework and Conflict Detection in O-RAN Near-RT RIC. *IEEE Communications Magazine*, 61(12):199–205, 2023. doi:10.1109/MCOM.018.2200752.

[3] Marcin Hoffmann, Salim Janji, Adam Samorzewski, Łukasz Kułacz, **C. Adamczyk**, Marcin Dryjański, Pawel Kryszkiewicz, Adrian Kliks, and Hanna Bogucka. Open RAN xApps Design and Evaluation: Lessons Learnt and Identified Challenges. *IEEE Journal on Selected Areas in Communications*, 42(2): 473–486, 2024. doi:10.1109/JSAC.2023.3336190.

[4] Abida Sultana, **C. Adamczyk**, Mayukh Roy Chowdhury, Adrian Kliks, and Aloizio Da Silva. Experimental evaluation of xApp Conflict Mitigation Framework in O-RAN: Insights from Testbed deployment in OTIC. In *IEEE INFOCOM 2025 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6, 2025. doi:10.1109/INFOCOMWKSHPS65812.2025.11152996.

[5] **C. Adamczyk** and Adrian Kliks. Detection and mitigation of indirect conflicts between xApps in Open Radio Access Networks. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–2, 2023. doi:10.1109/INFOCOMWKSHPS57453.2023.10225786.

[6] **C. Adamczyk** and Adrian Kliks. ACCoRD: Actor-Critic Conflict Resolution with Deep learning for O-RAN xApps, 2025. Preprint. doi:10.5281/zenodo.17752108.

[7] **C. Adamczyk** and Adrian Kliks. Wykrywanie konfliktów decyzji w inteligentnych sterownikach radiowych w otwartych sieciach dostępu radiowego O-RAN. *Przegląd Telekomunikacyjny - Wiadomości Telekomunikacyjne*, 4:404–407, 2022. ISSN 1230-3496. doi:10.15199/59.2022.4.68. URL `https://sigma-not.pl/publikacja-139000-wykrywanie-konflikt%C3%B3w-decyzji-w-inteligentnych-sterownikach-radiowych-w-otwartych-sieciach-dost%C4%99pu-radiowego-o-ran-przeglad-telekomunikacyjny-2022-4.html`.

Publication Summary

The following table provides an overview of the author's publications relevant to the thesis. The **Ref.** column corresponds to the citation number in the separate "Own Articles" bibliography list. Points assigned to relevant publication sources (i.e., journals, conferences, etc.) by the Polish Ministry of National Education are listed in column **MP**, according to the latest available list. Citation counts from Google Scholar (column **GS**) and IEEE Xplore (column **IEEE**) are current as of November 2025.

Ref.	Year	Article Title	Authors	MP*	GS	IEEE
[1]	2023	Challenges for Conflict Mitigation in O-RAN's RAN Intelligent Controllers	**C. Adamczyk**	20	12	6
[2]	2023	Conflict Mitigation Framework and Conflict Detection in O-RAN Near-RT RIC	**C. Adamczyk**, A. Kliks	200	42	25
[3]	2024	Open RAN xApps Design and Evaluation: Lessons Learnt and Identified Challenges	M. Hoffmann, S. Janji, A. Samorzewski, Ł. Kułacz, **C. Adamczyk**, M. Dryjański, P. Kryszkiewicz, A. Kliks, H. Bogucka	200	45	21
[4]	2025	Experimental evaluation of xApp Conflict Mitigation Framework in O-RAN: Insights from Testbed deployment in OTIC	A. Sultana, **C. Adamczyk**, M. R. Chowdhury, A. Kliks, A. Da Silva	200	0	0
[5]	2023	Detection and mitigation of indirect conflicts between xApps in Open Radio Access Networks	**C. Adamczyk**, A. Kliks	200	8	5
[6]	2025	ACCoRD: Actor-Critic Conflict Resolution with Deep learning for O-RAN xApps (preprint)**	**C. Adamczyk**, A. Kliks	-	0	0
[7]	2022	Wykrywanie konfliktów decyzji w inteligentnych sterownikach radiowych w otwartych sieciach dostępu radiowego O-RAN	**C. Adamczyk**, A. Kliks	20	0	0
			Total Citations		**107**	**57**

Table 8.1: List of the author's publications with ministry points and citation counts

Source: own elaboration based on Google Scholar and IEEE Xplore databases

**MP: Ministry points according to: Komunikat Ministra Nauki z dnia 05 stycznia 2024 r. w sprawie wykazu czasopism naukowych i recenzowanych materiałów z konferencji międzynarodowych.*

***Article currently under review for publication in The 4th Workshop on Next-generation Open and Programmable Radio Access Networks (NG-OPERA 2026), organized in conjunction with IEEE International Conference on Computer Communications 2026 (IEEE INFOCOM 2026).*

Bibliography

[8] Eurostat. Urban-rural Europe – Digital Society: Individuals – Devices used to access the internet, 2024. URL `https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Urban-rural_Europe_-_digital_society#Individuals_.E2.80.93_devices_used_to_access_the_internet`. Accessed: 2025-04-16.

[9] Pew Research Center. Americans' Use of Mobile Technology and Home Broadband, 2024. URL `https://www.pewresearch.org/internet/2024/01/31/americans-use-of-mobile-technology-and-home-broadband/`. Section "What share of Americans are constantly online?" Accessed: 2025-04-16.

[10] GSMA. The Mobile Economy Europe 2025, 2025. URL `https://www.gsma.com/solutions-and-impact/connectivity-for-good/mobile-economy/wp-content/uploads/2025/01/0125-Mobile-Economy-Europe-2025-web.pdf`. Accessed: 2025-04-17.

[11] GSMA. The Mobile Economy Europe 2022, 2022. URL `https://www.gsma.com/solutions-and-impact/connectivity-for-good/mobile-economy/wp-content/uploads/2024/05/ME-Europe-2022.pdf`. Accessed: 2025-04-17.

[12] GSMA. The Mobile Economy Europe 2023, 2023. URL `https://www.gsma.com/solutions-and-impact/connectivity-for-good/mobile-economy/wp-content/uploads/2023/11/GSMA-Mobile-Economy-Europe-2023.pdf`. Accessed: 2025-04-17.

[13] Jiacheng Chen, Xiaohu Liang, Jianzhe Xue, Yu Sun, Haibo Zhou, and Xuemin Shen. Evolution of RAN Architectures Toward 6G: Motivation, Development, and Enabling Technologies. *IEEE Communications Surveys & Tutorials*, 26(3):1950–1988, 2024. doi:10.1109/COMST.2024.3388511.

[14] Michele Polese, Mischa Dohler, Falko Dressler, Melike Erol-Kantarci, Rittwik Jana, Raymond Knopp, and Tommaso Melodia. Empowering the 6G Cellular Architecture With Open RAN. *IEEE Journal on Selected Areas in Communications*, 42(2):245–262, 2024. doi:10.1109/JSAC.2023.3334610.

[15] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges. *IEEE Communications Surveys & Tutorials*, 25(2):1376–1411, 2023. doi:10.1109/COMST.2023.3239220.

[16] O-RAN Alliance. O-RAN Alliance Official Website, 2025. URL `https://www.o-ran.org`. Accessed: 2025-11-26.

[17] O-RAN Working Group 3. *Near-Real-time RAN Intelligent Controller Conflict Mitigation*. O-RAN Alliance, October 2024. URL `https://www.o-ran.org/specifications`. v01.00. Accessed: 2025-11-26.

[18] Abdul Wadud, Fatemeh Golpayegani, and Nima Afraz. Conflict Management in the Near-RT-RIC of Open RAN: A Game Theoretic Approach. In *2023 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pages 479–486, 2023. doi:10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics60724.2023.00095.

[19] Anastasios E. Giannopoulos, Sotirios T. Spantideas, George Levis, Alexandros S. Kalafatelis, and Panagiotis Trakadas. COMIX: Generalized Conflict Management in O-RAN xApps—Architecture, Workflow, and a Power Control Case. *IEEE Access*, 13:116684–116700, 2025. doi:10.1109/ACCESS.2025.3585774.

[20] Hakan Erdol, Xiaoyang Wang, Robert Piechocki, George Oikonomou, and Arjun Parekh. xApp Distillation: AI-based Conflict Mitigation in B5G O-RAN. *Computer Networks*, 274:111848, 2026. ISSN 1389-1286. doi:10.1016/j.comnet.2025.111848. URL `https://www.sciencedirect.com/science/article/pii/S138912862500814X`.

[21] Pietro Brach del Prever, Salvatore D'Oro, Leonardo Bonati, Michele Polese, Maria Tsampazi, Heiko Lehmann, and Tommaso Melodia. PACIFISTA: Conflict Evaluation and Management in Open RAN. *IEEE Transactions on Mobile Computing*, pages 1–15, 2025. doi:10.1109/TMC.2025.3570632.

[22] Abdul Wadud, Fatemeh Golpayegani, and Nima Afraz. QACM: QoS-Aware xApp Conflict Mitigation in Open RAN. *IEEE Transactions on Green Communications and Networking*, 8(3): 978–993, 2024. doi:10.1109/TGCN.2024.3431945.

[23] Abdul Wadud, Fatemeh Golpayegani, and Nima Afraz. xApp-Level Conflict Mitigation in O-RAN, a Mobility Driven Energy Saving Case. In *IEEE INFOCOM 2025 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6, 2025. doi:10.1109/INFOCOMWKSHPS65812.2025.11153006.

[24] Idris Cinemre and Toktam Mahmoodi. xApp Conflict Mitigation with Scheduler, 2025. URL `https://arxiv.org/abs/2504.06867`. doi:10.48550/arXiv.2504.06867.

[25] J. Eberspächer, H.J. Vögel, C. Bettstetter, and C. Hartmann. *GSM - Architecture, Protocols and Services*. Wiley, 2008. ISBN 9780470741726. URL `https://books.google.pl/books?id=v6eN1tt9CEUC`.

[26] Martin Sauter. *From GSM to LTE: An Introduction to Mobile Networks and Mobile Broadband*. John Wiley & Sons, 2010. ISBN 9780470978238. doi:10.1002/9780470978238. URL `https://onlinelibrary.wiley.com/doi/book/10.1002/9780470978238`.

[27] B.H. Walke, P. Seidenberg, and M.P. Althoff. *UMTS: The Fundamentals*. Wiley, 2003. ISBN 9780470845578. doi:10.1002/0470014148. URL `https://onlinelibrary.wiley.com/doi/book/10.1002/0470014148`.

[28] KW Richardson. UMTS Overview. *Electronics & Communication Engineering Journal*, 12(3): 93–100, 2000.

[29] H. Holma and A. Toskala. *HSDPA/HSUPA for UMTS: High Speed Radio Access for Mobile Communications*. Wiley, 2007. ISBN 9780470032626. URL `https://books.google.pl/books?id=3kgImjyLBbIC`.

[30] Federico Boccardi, Robert W. Heath, Angel Lozano, Thomas L. Marzetta, and Petar Popovski. Five disruptive technology directions for 5G. *IEEE Communications Magazine*, 52(2):74–80, 2014. doi:10.1109/MCOM.2014.6736746.

[31] Ali Sufyan, Khan Bahadar Khan, Osama A. Khashan, Talha Mir, and Usama Mir. From 5G to beyond 5G: A Comprehensive Survey of Wireless Network Evolution, Challenges, and Promising Technologies. *Electronics*, 12(10), 2023. ISSN 2079-9292. doi:10.3390/electronics12102200. URL https://www.mdpi.com/2079-9292/12/10/2200.

[32] Shunliang Zhang. An Overview of Network Slicing for 5G. *IEEE Wireless Communications*, 26(3): 111–117, 2019. doi:10.1109/MWC.2019.1800234.

[33] *3GPP Technical Report TR 38.801 V14.0.0*. 3rd Generation Partnership Project (3GPP), March 2017. URL https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3056. Accessed: 2025-11-26.

[34] *3GPP Technical Specification TS 38.300 V18.5.0*. 3rd Generation Partnership Project (3GPP), March 2025. URL https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3191. Accessed: 2025-11-26.

[35] *3GPP Technical Specification TS 38.401 V18.1.0*. 3rd Generation Partnership Project (3GPP), March 2025. URL https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3219. Accessed: 2025-11-26.

[36] Marco Giordani, Michele Polese, Marco Mezzavilla, Sundeep Rangan, and Michele Zorzi. Toward 6G Networks: Use Cases and Technologies. *IEEE Communications Magazine*, 58(3):55–61, 2020. doi:10.1109/MCOM.001.1900411.

[37] S. Hämäläinen, H. Sanneck, and C. Sartori. *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency*. Wiley, 2012. ISBN 9781119970675. URL https://books.google.pl/books?id=qgLs2D5Hx8AC.

[38] *3GPP Technical Specification TS 28.313 V19.2.0*. 3rd Generation Partnership Project (3GPP), March 2025. URL https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3695. Accessed: 2025-11-26.

[39] Mahmoud A. Hasabelnaby, Mohanad Obeed, Mohammed Saif, Anas Chaaban, and M. J. Hossain. From Centralized RAN to Open RAN: A Survey on the Evolution of Distributed Antenna Systems, 2024. URL https://arxiv.org/abs/2411.12166. doi:10.48550/arXiv.2411.12166.

[40] Mihia Kassi and Soumaya Hamouda. RAN Virtualization: How Hard Is It to Fully Achieve? *IEEE Access*, 12:38030–38047, 2024. doi:10.1109/ACCESS.2024.3375749.

[41] Aleksandra Checko, Henrik L. Christiansen, Ying Yan, Lara Scolari, Georgios Kardaras, Michael S. Berger, and Lars Dittmann. Cloud RAN for Mobile Networks—A Technology Overview. *IEEE Communications Surveys & Tutorials*, 17(1):405–426, 2015. doi:10.1109/COMST.2014.2355255.

[42] Prabhu Kaliyammal Thiruvasagam, Chandrasekar T, Vinay Venkataram, Vivek Raja Ilangovan, Maneesha Perapalla, Rajisha Payyanur, Senthilnathan M D, Vishal Kumar, and Kokila J. Open RAN: Evolution of Architecture, Deployment Aspects, and Future Directions, 2023. URL https://arxiv.org/abs/2301.06713. doi:10.48550/arXiv.2301.06713.

[43] O-RAN Working Group 1. *O-RAN Architecture Description*. O-RAN Alliance, October 2025. URL https://www.o-ran.org/specifications. v15.00. Accessed: 2025-11-26.

[44] O-RAN Working Group 2. *Non-Real-time RAN Intelligent Controller Architecture*. O-RAN Alliance, June 2025. URL https://www.o-ran.org/specifications. v07.00. Accessed: 2025-11-26.

[45] O-RAN Working Group 3. *Near-Real-time RAN Intelligent Controller Architecture*. O-RAN Alliance, February 2025. URL https://www.o-ran.org/specifications. v07.00. Accessed: 2025-11-26.

[46] O-RAN Working Group 2. *Non-RT RIC & A1/R1 interface: Use Cases and Requirements*. O-RAN Alliance, October 2025. URL `https://www.o-ran.org/specifications`. v10.04. Accessed: 2025-11-26.

[47] O-RAN Working Group 10. *O-RAN Operations and Maintenance Architecture*. O-RAN Alliance, June 2025. URL `https://www.o-ran.org/specifications`. v15.00. Accessed: 2025-11-26.

[48] O-RAN Working Group 10. *O-RAN O1 Interface*. O-RAN Alliance, June 2025. URL `https://www.o-ran.org/specifications`. v16.00. Accessed: 2025-11-26.

[49] O-RAN Working Group 3. *Near-Real-time RAN Intelligent Controller Use Cases and Requirements*. O-RAN Alliance, October 2025. URL `https://www.o-ran.org/specifications`. v09.00. Accessed: 2025-11-26.

[50] O-RAN Working Group 3. *Near-Real-time RAN Intelligent Controller, E2 Service Model (E2SM)*. O-RAN Alliance, February 2025. URL `https://www.o-ran.org/specifications`. v07.00. Accessed: 2025-11-26.

[51] Simona Marinova and Alberto Leon-Garcia. Intelligent O-RAN Beyond 5G: Architecture, Use Cases, Challenges, and Opportunities. *IEEE Access*, 12:27088–27114, 2024. doi:10.1109/ACCESS.2024.3367289.

[52] Salvatore D'Oro, Michele Polese, Leonardo Bonati, Hai Cheng, and Tommaso Melodia. dApps: Distributed Applications for Real-Time Inference and Control in O-RAN. *IEEE Communications Magazine*, 60(11):52–58, 2022. doi:10.1109/MCOM.002.2200079.

[53] Andrea Lacava, Leonardo Bonati, Niloofar Mohamadi, Rajeev Gangula, Florian Kaltenberger, Pedram Johari, Salvatore D'Oro, Francesca Cuomo, Michele Polese, and Tommaso Melodia. dApps: Enabling Real-Time AI-Based Open RAN Control. *Computer Networks*, 269:111342, 2025. ISSN 1389-1286. doi:10.1016/j.comnet.2025.111342. URL `https://www.sciencedirect.com/science/article/pii/S1389128625003093`.

[54] O-RAN Working Group 2. *R1 interface: Use Cases and Requirements*. O-RAN Alliance, October 2025. URL `https://www.o-ran.org/specifications`. v11.00. Accessed: 2025-11-26.

[55] O-RAN Working Group 3. *Near-Real-time RAN Intelligent Controller, RIC APIs*. O-RAN Alliance, June 2024. URL `https://www.o-ran.org/specifications`. v02.00. Accessed: 2025-11-26.

[56] O-RAN Working Group 3. *Near-Real-time RAN Intelligent Controller, E2 General Aspects and Principles (E2GAP)*. O-RAN Alliance, October 2025. URL `https://www.o-ran.org/specifications`. v08.00. Accessed: 2025-11-26.

[57] O-RAN Working Group 3. *Near-Real-time RAN Intelligent Controller, E2 Application Protocol (E2AP)*. O-RAN Alliance, October 2025. URL `https://www.o-ran.org/specifications`. v08.00. Accessed: 2025-11-26.

[58] O-RAN Working Group 3. *Near-Real-time RAN Intelligent Controller, E2 Service Model (E2SM), RAN Control*. O-RAN Alliance, October 2025. URL `https://www.o-ran.org/specifications`. v09.00. Accessed: 2025-11-26.

[59] O-RAN Working Group 3. *Near-Real-time RAN Intelligent Controller, E2 Service Model (E2SM), RAN Function Network Interface (NI)*. O-RAN Alliance, February 2020. URL `https://www.o-ran.org/specifications`. v01.00. Accessed: 2025-11-26.

[60] O-RAN Working Group 3. *Near-Real-time RAN Intelligent Controller, E2 Service Model (E2SM), Cell Configuration and Control (CCC)*. O-RAN Alliance, June 2025. URL `https://www.o-ran.org/specifications`. v06.00. Accessed: 2025-11-26.

[61] O-RAN Working Group 3. *Near-Real-time RAN Intelligent Controller, E2 Service Model (E2SM), Lower Layers Control (LLC)*. O-RAN Alliance, February 2025. URL `https://www.o-ran.org/specifications`. v01.00. Accessed: 2025-11-26.

[62] O-RAN Working Group 3. *Near-Real-time RAN Intelligent Controller, E2 Service Model (E2SM), Key Performance Monitoring (KPM)*. O-RAN Alliance, October 2025. URL `https://www.o-ran.org/specifications`. v07.00. Accessed: 2025-11-26.

[63] O-RAN Working Group 2. *A1 interface: Use Cases and Requirements*. O-RAN Alliance, June 2025. URL `https://www.o-ran.org/specifications`. v03.00. Accessed: 2025-11-26.

[64] O-RAN next Generation Research Group (nGRG). *dApps for Real-Time RAN Control: Use Cases and Requirements*. O-RAN next Generation Research Group (nGRG), October 2024. URL `https://www.o-ran.org/research-reports/dapps-for-real-time-ran-control-use-cases-and-requirements`. Report ID: RR-2024-10. Accessed: 26-11-2025.

[65] Arman Elyasi, Andrew Ashdown, KM Rumman, and Francesco Restuccia. O-RAN xApps: Survey and Research Challenges. *preprint*, 01 2025. doi:10.2139/ssrn.5236117. URL `https://ssrn.com/abstract=5236117`.

[66] Chafika Benzaïd and Tarik Taleb. AI for Beyond 5G Networks: A Cyber-Security Defense or Offense Enabler? *IEEE Network*, 34(6):140–147, 2020. doi:10.1109/MNET.011.2000088.

[67] Bouziane Brik, Hatim Chergui, Lanfranco Zanzi, Francesco Devoti, Adlen Ksentini, Muhammad Shuaib Siddiqui, Xavier Costa-Pèrez, and Christos Verikoukis. Explainable AI in 6G O-RAN: A Tutorial and Survey on Architecture, Use Cases, Challenges, and Future Research. *IEEE Communications Surveys & Tutorials*, pages 1–1, 2024. doi:10.1109/COMST.2024.3510543.

[68] O-RAN Working Group 2. *AI/ML workflow description and requirements*. O-RAN Alliance, October 2021. URL `https://www.o-ran.org/specifications`. v01.03. Accessed: 2025-11-26.

[69] Paul H. Masur, Jeffrey H. Reed, and Nishith K. Tripathi. Artificial Intelligence in Open-Radio Access Network. *IEEE Aerospace and Electronic Systems Magazine*, 37(9):6–15, 2022. doi:10.1109/MAES.2022.3186966.

[70] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

[71] Mutasem Q. Hamdan, Haeyoung Lee, Dionysia Triantafyllopoulou, Rúben Borralho, Abdulkadir Kose, Esmaeil Amiri, David Mulvey, Wenjuan Yu, Rafik Zitouni, Riccardo Pozza, Bernie Hunt, Hamidreza Bagheri, Chuan Heng Foh, Fabien Heliot, Gaojie Chen, Pei Xiao, Ning Wang, and Rahim Tafazolli. Recent Advances in Machine Learning for Network Automation in the O-RAN. *Sensors*, 23 (21), 2023. ISSN 1424-8220. doi:10.3390/s23218792. URL `https://www.mdpi.com/1424-8220/23/21/8792`.

[72] Anastasios Giannopoulos, Sotirios Spantideas, Nikolaos Kapsalis, Panagiotis Gkonis, Lambros Sarakis, Christos Capsalis, Massimo Vecchio, and Panagiotis Trakadas. Supporting Intelligence in Disaggregated Open Radio Access Networks: Architectural Principles, AI/ML Workflow, and Use Cases. *IEEE Access*, 10:39580–39595, 2022. doi:10.1109/ACCESS.2022.3166160.

[73] Xingqin Lin, Lopamudra Kundu, Chris Dick, and Soma Velayutham. Embracing AI in 5G-Advanced Toward 6G: A Joint 3GPP and O-RAN Perspective. *IEEE Communications Standards Magazine*, 7 (4):76–83, 2023. doi:10.1109/MCOMSTD.0005.2200070.

[74] Hoejoo Lee, Jiwon Cha, Daeken Kwon, Myeonggi Jeong, and Intaik Park. Hosting AI/ML Workflows on O-RAN RIC Platform. In *2020 IEEE Globecom Workshops (GC Wkshps*, pages 1–6, 2020. doi:10.1109/GCWkshps50303.2020.9367572.

[75] *Acumos Documentation*. Acumos AI Project, 2025. URL `https://docs.acumos.org/en/latest/`. Accessed: 2025-11-26.

[76] Rahul Banerji, Naman Gupta, Suman Kumar, Sukhdeep Singh, Avinash Bhat, Bharat J. R. Sahu, and Seungil Yoon. Onap based pro-active access discovery and selection for 5g networks. In *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–6, 2020. doi:10.1109/WCNCW48565.2020.9124724.

[77] O-RAN Software Community. O-RAN Software Community Projects: Near Realtime RAN Intelligent Controller (RIC), 2025. URL `https://docs.o-ran-sc.org/en/latest/projects.html#near-realtime-ran-intelligent-controller-ric`. Accessed: 2025-11-26.

[78] Mohammad Asif Habibi, Bin Han, Merve Saimler, Ignacio Labrador Pavon, and Hans D. Schotten. Towards an AI/ML-driven SMO Framework in O-RAN: Scenarios, Solutions, and Challenges, 2024. URL `https://arxiv.org/abs/2409.05092`. doi:10.48550/arXiv.2409.05092.

[79] Yaser Azimi, Saleh Yousefi, Hashem Kalbkhani, and Thomas Kunz. Applications of Machine Learning in Resource Management for RAN-Slicing in 5G and Beyond Networks: A Survey. *IEEE Access*, 10:106581–106612, 2022. doi:10.1109/ACCESS.2022.3210254.

[80] Leonardo Bonati, Salvatore D'Oro, Michele Polese, Stefano Basagni, and Tommaso Melodia. Intelligence and Learning in O-RAN for Data-Driven NextG Cellular Networks. *IEEE Communications Magazine*, 59(10):21–27, 2021. doi:10.1109/MCOM.101.2001120.

[81] Leonardo Bonati, Pedram Johari, Michele Polese, Salvatore D'Oro, Subhramoy Mohanti, Miead Tehrani-Moayyed, Davide Villa, Shweta Shrivastava, Chinenye Tassie, Kurt Yoder, Ajeet Bagga, Paresh Patel, Ventz Petkov, Michael Seltser, Francesco Restuccia, Abhimanyu Gosain, Kaushik R. Chowdhury, Stefano Basagni, and Tommaso Melodia. Colosseum: Large-Scale Wireless Experimentation Through Hardware-in-the-Loop Network Emulation. In *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 105–113, 2021. doi:10.1109/DySPAN53946.2021.9677430.

[82] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Pedram Johari, Davide Villa, Sakthivel Velumani, Rajeev Gangula, Maria Tsampazi, Clifton Paul Robinson, Gabriele Gemmi, Andrea Lacava, Stefano Maxenti, Hai Cheng, and Tommaso Melodia. Colosseum: The Open RAN Digital Twin. *IEEE Open Journal of the Communications Society*, 5:5452–5466, 2024. doi:10.1109/OJCOMS.2024.3447472.

[83] Maria Tsampazi, Salvatore D'Oro, Michele Polese, Leonardo Bonati, Gwenael Poitau, Michael Healy, and Tommaso Melodia. A Comparative Analysis of Deep Reinforcement Learning-Based xApps in O-RAN. In *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, pages 1638–1643, 2023. doi:10.1109/GLOBECOM54140.2023.10437367.

[84] Hoejoo Lee, Youngcheol Jang, Juhwan Song, and Hunje Yeon. O-RAN AI/ML Workflow Implementation of Personalized Network Optimization via Reinforcement Learning. In *2021 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, 2021. doi:10.1109/GCWkshps52748.2021.9681936.

[85] Mohammadreza Kouchaki and Vuk Marojevic. Actor-Critic Network for O-RAN Resource Allocation: xApp Design, Deployment, and Analysis. In *2022 IEEE Globecom Workshops (GC Wkshps)*, pages 968–973, 2022. doi:10.1109/GCWkshps56602.2022.10008713.

[86] Ryan Barker, Alireza Ebrahimi Dorcheh, Tolunay Seyfi, and Fatemeh Afghah. REAL: Reinforcement Learning-Enabled xApps for Experimental Closed-Loop Optimization in O-RAN with OSC RIC and

srsRAN. In *2025 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 389–395, 06 2025. doi:10.1109/ICCWorkshops67674.2025.11162144.

[87] Md Arafat Habib, Hao Zhou, Pedro Enrique Iturria-Rivera, Medhat Elsayed, Majid Bavand, Raimundas Gaigalas, Yigit Ozcan, and Melike Erol-Kantarci. Intent-driven Intelligent Control and Orchestration in O-RAN Via Hierarchical Reinforcement Learning. In *2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, pages 55–61, 2023. doi:10.1109/MASS58611.2023.00015.

[88] Md Arafat Habib, Hao Zhou, Pedro Enrique Iturria-Rivera, Yigit Ozcan, Medhat Elsayed, Majid Bavand, Raimundas Gaigalas, and Melike Erol-Kantarci. Machine Learning-Enabled Traffic Steering in O-RAN: A Case Study on Hierarchical Learning Approach. *IEEE Communications Magazine*, 63 (1):100–107, 2025. doi:10.1109/MCOM.002.2300526.

[89] Oner Orhan, Vasuki Narasimha Swamy, Thomas Tetzlaff, Marcel Nassar, Hosein Nikopour, and Shilpa Talwar. Connection Management xApp for O-RAN RIC: A Graph Neural Network and Reinforcement Learning Approach. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 936–941, 2021. doi:10.1109/ICMLA52953.2021.00154.

[90] Han Zhang, Hao Zhou, and Melike Erol-Kantarci. Team Learning-Based Resource Allocation for Open Radio Access Network (O-RAN). In *ICC 2022 - IEEE International Conference on Communications*, pages 4938–4943, 2022. doi:10.1109/ICC45855.2022.9838763.

[91] Pedro Enrique Iturria-Rivera, Han Zhang, Hao Zhou, Shahram Mollahasani, and Melike Erol-Kantarci. Multi-Agent Team Learning in Virtualized Open Radio Access Networks (O-RAN). *Sensors*, 22(14), 2022. ISSN 1424-8220. doi:10.3390/s22145375. URL `https://www.mdpi.com/1424-8220/22/14/5375`.

[92] Ahmad Ahmadi Siahpoush and Vahid Shah-Mansouri. Distributed Deep Reinforcement Learning for Radio Resource Management in O-RAN. In *2024 32nd International Conference on Electrical Engineering (ICEE)*, pages 1–7, 2024. doi:10.1109/ICEE63041.2024.10667952.

[93] Farhad Rezazadeh, Lanfranco Zanzi, Francesco Devoti, Sergio Barrachina-Muñoz, Engin Zeydan, Xavier Costa-Pérez, and Josep Mangues-Bafalluy. A Multi-Agent Deep Reinforcement Learning Approach for RAN Resource Allocation in O-RAN. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–2, 2023. doi:10.1109/INFOCOMWKSHPS57453.2023.10226154.

[94] Payam Abdisarabshali, Nicholas Accurso, Filippo Malandra, Weifeng Su, and Seyyedali Hosseinalipour. Elastic Federated Learning over Open Radio Access Network (O-RAN) for Concurrent Execution of Multiple Distributed Learning Tasks, 2025. URL `https://arxiv.org/abs/2305.02109`. doi:10.48550/arXiv.2305.02109.

[95] Han Zhang, Hao Zhou, and Melike Erol-Kantarci. Federated Deep Reinforcement Learning for Resource Allocation in O-RAN Slicing. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pages 958–963, 2022. doi:10.1109/GLOBECOM48099.2022.10001658.

[96] Muhammad Asad and Safa Otoum. Federated learning for efficient spectrum allocation in open RAN. *Cluster Computing*, 27(8):11237–11247, Nov 2024. ISSN 1573-7543. doi:10.1007/s10586-024-04500-9.

[97] Hakan Erdol, Xiaoyang Wang, Peizheng Li, Jonathan D. Thomas, Robert Piechocki, George Oikonomou, Rui Inacio, Abdelrahim Ahmad, Keith Briggs, and Shipra Kapoor. Federated Meta-Learning for Traffic Steering in O-RAN. In *2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)*, pages 1–7, 2022. doi:10.1109/VTC2022-Fall57202.2022.10012789.

[98] Mohammadreza Kouchaki, Aly Sabri Abdalla, and Vuk Marojevic. OpenAI dApp: An Open AI Platform for Distributed Federated Reinforcement Learning Apps in O-RAN. In *2023 IEEE Future Networks World Forum (FNWF)*, pages 1–6, 2023. doi:10.1109/FNWF58287.2023.10520642.

[99] Xingqi Wu, Yuhui Wang, Junaid Farooq, and Juntao Chen. LLM-Driven Agentic AI Approach to Enhanced O-RAN Resilience in Next-Generation Networks, March 2025. URL `http://dx.doi.org/10.36227/techrxiv.174284755.59863143/v1`. doi:10.36227/techrxiv.174284755.59863143/v1.

[100] Xingqi Wu, Junaid Farooq, Yuhui Wang, and Juntao Chen. LLM-xApp: A Large Language Model Empowered Radio Resource Management xApp for 5G O-RAN. In *Symposium on Networks and Distributed Systems Security (NDSS), Workshop on Security and Privacy of Next-Generation Networks (FutureG 2025), San Diego, CA*, 01 2025. doi:10.14722/futureg.2025.23057.

[101] Mohammed M. H. Qazzaz, Łukasz Kułacz, Adrian Kliks, Syed A. Zaidi, Marcin Dryjanski, and Des McLernon. Machine Learning-based xApp for Dynamic Resource Allocation in O-RAN Networks. In *2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*, pages 492–497, 2024. doi:10.1109/ICMLCN59089.2024.10625184.

[102] Leonardo Bonati, Michele Polese, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. OpenRAN Gym: An Open Toolbox for Data Collection and Experimentation with AI in O-RAN. In *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 518–523, 2022. doi:10.1109/WCNC51071.2022.9771908.

[103] Leonardo Bonati, Michele Polese, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. OpenRAN Gym: AI/ML development, data collection, and testing for O-RAN on PAWR platforms. *Computer Networks*, 220:109502, 2023. ISSN 1389-1286. doi:10.1016/j.comnet.2022.109502. URL `https://www.sciencedirect.com/science/article/pii/S1389128622005369`.

[104] AI-RAN Alliance. AI-RAN Alliance Vision and Mission White Paper, 12 2024. URL `https://ai-ran.org/wp-content/uploads/2024/12/AI-RAN_Alliance_Whitepaper.pdf`. Accessed: 2025-05-03.

[105] Nvidia. AI-RAN: Artificial Intelligent - Radio Access Networks. Frequently Asked Questions, March 2025. URL `https://docs.nvidia.com/aerial-resources/2025_AI-RAN_FAQ.pdf`. Accessed: 2025-05-03.

[106] SoftBank. AI-RAN: Telecom Infrastructure for the Age of AI, December 2024. URL `https://www.softbank.jp/corp/set/data/technology/research/story-event/Whitepaper_Download_Location/pdf/SoftBank_AI_RAN_Whitepaper_December2024.pdf`. Accessed: 2025-05-05.

[107] Noe M. Yungaicela-Naula, Vishal Sharma, and Sandra Scott-Hayward. Misconfiguration in O-RAN: Analysis of the impact of AI/ML. *Computer Networks*, 247:110455, 2024. ISSN 1389-1286. doi:10.1016/j.comnet.2024.110455. URL `https://www.sciencedirect.com/science/article/pii/S1389128624002871`.

[108] O-RAN Working Group 2. *A1 interface: Type Definitions*. O-RAN Alliance, October 2025. URL `https://www.o-ran.org/specifications`. v11.00. Accessed: 2025-11-26.

[109] Anubhab Banerjee, Stephen S. Mwanje, and Georg Carle. Game theoretic conflict resolution mechanism for cognitive autonomous networks. In *Proceedings of the 2020 Summer Simulation Conference*, SummerSim '20, San Diego, CA, USA, 2020. Society for Computer Simulation International. ISBN 9781713814290. doi:10.5555/3427510.3427548.

[110] O-RAN Working Group 2. *A1 interface: Application Protocol*. O-RAN Alliance, February 2025. URL `https://www.o-ran.org/specifications`. v04.04. Accessed: 2025-11-26.

[111] O-RAN Working Group 2. *Non-Real-time RIC and A1 interface: Study on A1 policy conflict mitigation*. O-RAN Alliance, June 2025. URL `https://www.o-ran.org/specifications`. v01.00. Accessed: 2025-11-26.

[112] C. Prehofer and C. Bettstetter. Self-organization in communication networks: principles and design paradigms. *IEEE Communications Magazine*, 43(7):78–85, 2005. doi:10.1109/MCOM.2005.1470824.

[113] Martin Döttling and Ingo Viering. Challenges in mobile network operation: Towards self-optimizing networks. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3609–3612, 2009. doi:10.1109/ICASSP.2009.4960407.

[114] Hafiz Yasar Lateef, Ali Imran, and Adnan Abu-dayya. A framework for classification of Self-Organising network conflicts and coordination algorithms. In *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 2898–2903, 2013. doi:10.1109/PIMRC.2013.6666642.

[115] Hafiz Yasar Lateef, Ali Imran, Muhammad Ali Imran, Lorenza Giupponi, and Mischa Dohler. LTE-advanced self-organizing network conflicts and coordination algorithms. *IEEE Wireless Communications*, 22(3):108–117, 2015. doi:10.1109/MWC.2015.7143333.

[116] Xavier Gelabert, Berna Sayrac, and Sana Ben Jemaa. A performance evaluation framework for control loop interaction in Self Organizing Networks. In *2011 IEEE 22nd International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 263–267, 2011. doi:10.1109/PIMRC.2011.6139962.

[117] Lars Christoph Schmelz, Mehdi Amirijoo, Andreas Eisenblaetter, Remco Litjens, Michaela Neuland, and John Turk. A coordination framework for self-organisation in LTE networks. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pages 193–200, 2011. doi:10.1109/INM.2011.5990691.

[118] Ovidiu Iacoboaiea, Berna Sayrac, Sana Ben Jemaa, and Pascal Bianchi. SON conflict diagnosis in heterogeneous networks. In *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1459–1463, 2015. doi:10.1109/PIMRC.2015.7343528.

[119] Abdoulaye Tall, Richard Combes, Zwi Altman, and Eitan Altman. Distributed Coordination of Self-Organizing Mechanisms in Communication Networks. *IEEE Transactions on Control of Network Systems*, 1(4):328–337, 2014. doi:10.1109/TCNS.2014.2357511.

[120] Tanmoy Bag, Sharva Garg, Sriram Parameswaran, Diego Preciado, and Andreas Mitschele-Thiel. Communication-Efficient and Scalable Management of Self-Organizing Mobile Networks. In *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, pages 1–5, 2023. doi:10.1109/NOMS56928.2023.10154274.

[121] Tobias Bandh, Henning Sanneck, and Raphael Romeikat. An Experimental System for SON Function Coordination. In *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, pages 1–2, 2011. doi:10.1109/VETECS.2011.5956708.

[122] Raphael Romeikat, Bernhard Bauer, Tobias Bandh, Georg Carle, Henning Sanneck, and Lars Schmelz. Policy-driven workflows for mobile network management automation. In *IWCMC '10: Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, pages 1111–1115, 11 2010. doi:10.1145/1815396.1815650.

[123] Tobias Bandh, Raphael Romeikat, Henning Sanneck, and Haitao Tang. Policy-based coordination and management of SON functions. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pages 827–840, 2011. doi:10.1109/INM.2011.5990492.

[124] Kostas Tsagkaris, Nikos Koutsouris, Panagiotis Demestichas, Richard Combes, and Zwi Altman. SON Coordination in a Unified Management Framework. In *2013 IEEE 77th Vehicular Technology Conference (VTC Spring)*, pages 1–5, 2013. doi:10.1109/VTCSpring.2013.6692759.

[125] N. Koutsouris, K. Tsagkaris, P. Demestichas, Z. Altman, R. Combes, P. Peloso, L. Ciavaglia, L. Mamatas, S. Clayman, and A. Galis. Conflict free coordination of SON functions in a Unified Management Framework: Demonstration of a proof of concept prototyping platform. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 1092–1093, 2013. ISBN 978-1-4673-5229-1.

[126] Idris Cinemre, Kashif Mehmood, Katina Kralevska, and Toktam Mahmoodi. Direct-Conflict Resolution in Intent-Driven Autonomous Networks. In *European Wireless 2023; 28th European Wireless Conference*, pages 142–147, 2023. ISBN 9783800762255. doi:10.48550/arXiv.2401.08341.

[127] Ayhan Akbas, Chuan Heng Foh, Hamed Alimohammadi, Ahmad Sulaymani Seyed, Samara Mayhoub, Sulyman Abdulkareem, Chee Yen Leow, and Mohammad Sojafar. Coverage Versus Capacity Conflict Mediation via Antenna Tilt in Self-Organising Networks. In *2024 IEEE 7th International Symposium on Telecommunication Technologies (ISTT)*, pages 72–77, 2024. doi:10.1109/ISTT63363.2024.10750758.

[128] Janne Ali-Tolppa and Tsvetko Tsvetkov. Optimistic concurrency control in self-organizing networks using automatic coordination and verification. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 618–624, 2016. doi:10.1109/NOMS.2016.7502867.

[129] Zhiqiang Liu, Peilin Hong, Kaiping Xue, and Min Peng. Conflict Avoidance between Mobility Robustness Optimization and Mobility Load Balancing. In *2010 IEEE Global Telecommunications Conference (GLOBECOM)*, 2010. doi:10.1109/GLOCOM.2010.5683861.

[130] Miaona Huang and Jun Chen. A conflict avoidance scheme between mobility load balancing and mobility robustness optimization in self-organizing networks. *Wireless Networks*, 24(1):271–281, 2018. doi:10.1007/s11276-016-1331-y.

[131] Yun Li, Man Li, Bin Cao, and Wenjing Liu. A conflict avoid method between Load Balancing and Mobility Robustness Optimization in LTE. In *2012 1st IEEE International Conference on Communications in China (ICCC)*, pages 143–148, 2012. doi:10.1109/ICCChina.2012.6356868.

[132] Nauman Zia, Stephen S. Mwanje, and Andreas Mitschele-Thiel. A policy based conflict resolution mechanism for MLB and MRO in LTE self-optimizing networks. In *2014 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6, 2014. doi:10.1109/ISCC.2014.6912543.

[133] Pablo Muñoz, Raquel Barco, and Sergio Fortes. Conflict Resolution Between Load Balancing and Handover Optimization in LTE Networks. *IEEE Communications Letters*, 18(10):1795–1798, 2014. doi:10.1109/LCOMM.2014.2350473.

[134] Yi-Hao Tu, Yi-Wei Ma, Zhi-Xiang Li, Jiann-Liang Chen, and Kazuya Tsukamoto. Applying Deep Reinforcement Learning for Self-organizing Network Architecture. In *2023 IEEE 6th International Conference on Knowledge Innovation and Invention (ICKII)*, pages 16–20, 2023. doi:10.1109/ICKII58656.2023.10332666.

[135] Gerasimos Stamatelatos, Aggeliki Sgora, and Nancy Alonistioti. Intelligent SON Coordination in the 5G-andbeyond era. In *2022 Global Information Infrastructure and Networking Symposium (GIIS)*, pages 99–103, 2022. doi:10.1109/GIIS56506.2022.9936918.

[136] Ovidiu Iacoboaiea, Berna Sayrac, Sana Ben Jemaa, and Pascal Bianchi. SON Coordination for parameter conflict resolution: A reinforcement learning framework. In *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 196–201, 2014. doi:10.1109/WCNCW.2014.6934885.

[137] Stephen S. Mwanje and Andreas Mitschele-Thiel. Concurrent cooperative games for coordinating SON functions in cognitive cellular networks. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1298–1303, 2015. doi:10.1109/INM.2015.7140485.

[138] Jessica Moysen, Mario Garcia-Lozano, Lorenza Giupponi, and Silvia Ruiz. Conflict Resolution in Mobile Networks: A Self-Coordination Framework Based on Non-Dominated Solutions and Machine Learning for Data Analytics [Application Notes]. *IEEE Computational Intelligence Magazine*, 13(2): 52–64, 2018. doi:10.1109/MCI.2018.2807038.

[139] Diego Fernando Preciado Rojas, Faiaz Nazmetdinov, and Andreas Mitschele-Thiel. Zero-touch coordination framework for Self-Organizing Functions in 5G. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–8, 2020. doi:10.1109/WCNC45663.2020.9120799.

[140] Eunsok Lee, Kihoon Kim, Subin Han, and Sangheon Pack. A Scalable and Low-Complexity Coordination Framework for Self-Organizing Networks. In *2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall)*, pages 1–5, 2024. doi:10.1109/VTC2024-Fall63153.2024.10757566.

[141] Panagiotis Vlacheas, Evangelos Thomatos, Kostas Tsagkaris, and Panagiotis Demestichas. Operator-governed SON coordination in downlink LTE networks. In *2012 Future Network & Mobile Summit (FutureNetw)*, pages 1–9, 2012. ISBN 9781467303200.

[142] Arshia Zolghadr, Joao F. Santos, Luiz A. DaSilva, and Jacek Kibilda. Learning and Reconstructing Conflicts in O-RAN: A Graph Neural Network Approach. In *2025 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 01–06, 2025. doi:10.1109/WCNC61545.2025.10978358.

[143] Hammad Zafar, Ehsan Tohidi, Martin Kasparick, and Sławomir Stańczak. Conflict Mitigation Approach for O-RAN xApps. In *2025 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2025. doi:10.1109/WCNC61545.2025.10978767.

[144] Joss Armstrong, Enda Fallon, and Sheila Fallon. Pre-Emptive Conflict Detection Architecture for O-RAN Service Management and Orchestration. In *2024 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, pages 335–340, 2024. doi:10.1109/IAICT62357.2024.10617647.

[145] Maryam Al Shami, Jun Yan, and Emmanuel Thepie Fapi. O-RAN xApps Conflict Management using Graph Convolutional Networks, 2025. URL `https://arxiv.org/abs/2503.03523`. doi:10.48550/arXiv.2503.03523.

[146] Arshia Zolghadr. Learning and Reconstructing Conflicts in O-RAN. Master's thesis, Virginia Polytechnic Institute and State University, Arlington, VA, 04 2025. URL `https://hdl.handle.net/10919/134963`. Accessed: 2025-11-26.

[147] Pragya Sharma, Shihua Sun, Shachi Deshpande, Angelos Stavrou, and Haining Wang. Towards xApp Conflict Evaluation with Explainable Machine Learning and Causal Inference in O-RAN, 2025. URL `https://arxiv.org/abs/2510.13031`.

[148] Abida Sultana, Fahim Bashar, Mayukh Roy Chowdhury, and Aloizio Pereira Da Silva. A Software-Defined Radio based O-RAN Platform for xApp Conflict Detection and Mitigation. In *IEEE Military Communications Conference (MILCOM)*, pages 686–687, 2024. doi:10.1109/MILCOM61039.2024.10773739.

[149] Qiushuo Hou, Sangwoo Park, Matteo Zecchin, Yunlong Cai, Guanding Yu, and Osvaldo Simeone. What If We Had Used a Different App? Reliable Counterfactual KPI Analysis in Wireless Systems. *IEEE Transactions on Cognitive Communications and Networking*, pages 1–1, 2025. doi:10.1109/TCCN.2025.3536793.

[150] Marius Corici, Ramona Modroiu, Fabian Eichhorn, Eric Troudt, and Thomas Magedanz. Towards efficient conflict mitigation in the converged 6G Open RAN control plane. *Annals of Telecommunications*, 79(9):621–631, Oct 2024. ISSN 1958-9395. doi:10.1007/s12243-024-01036-2. URL `https://doi.org/10.1007/s12243-024-01036-2`.

[151] ns-3 Consortium / nsnam.org. ns-3: a discrete-event network simulator for Internet systems, 2025. URL `https://www.nsnam.org/`. Accessed: 2025-10-16.

[152] Andrea Lacava, Michele Polese, Rajarajan Sivaraj, Rahul Soundrarajan, Bhawani Shanker Bhati, Tarunjeet Singh, Tommaso Zugno, Francesca Cuomo, and Tommaso Melodia. Programmable and Customized Intelligence for Traffic Steering in 5G Networks Using Open RAN Architectures. *IEEE Transactions on Mobile Computing*, pages 1–16, 2023. ISSN 1558-0660. doi:10.1109/TMC.2023.3266642.

[153] OMNeT++. OMNeT++ Discrete Event Simulator, 2025. URL `https://omnetpp.org/`. Accessed: 2025-10-16.

[154] Dieter J. Cichon, Thomas Kürner, Peter J. Cullen, Jean-Frédéric Wagen, Jan-Erik Berg, Jaakko Lähteenmäki, Bernard Fleury, and Ernst Bonek. Propagation Prediction Models. In Eraldo Damosso and Luis M. Correia, editors, *COST Action 231 : Digital mobile radio towards future generation systems: Final Report*, EUR 18957, chapter 4, pages 115–208. European Commission, Brussels, Belgium, 1999. ISBN 9282854167. URL `https://www.winlab.rutgers.edu/~andrej/research/docs/cost231/ch4.pdf`. Accessed: 2025-11-26.

[155] 3GPP. 3GPP technical report TR 38.901 v16.1.0. Technical Report TR 38 901 V16.1.0, 3GPP, November 2020. URL `https://www.3gpp.org/dynareport/38901.htm`. Accessed: 2025-11-26.

[156] Pedro Casas-Hernandez, Pierdomenico Fiadino, and Alessandro D´Alconzo. Machine-Learning Based Approaches for Anomaly Detection and Classification in Cellular Networks. In *Proceedings of the 8th International Workshop on Traffic Monitoring and Analysis*, pages 1–8, 2016. URL `https://tma.ifip.org/2016/papers/tma2016-final50.pdf`. Accessed: 2025-11-26.

[157] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018. URL `https://arxiv.org/abs/1506.02438`. doi:10.48550/arXiv.1506.02438.

[158] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017. URL `https://arxiv.org/abs/1707.06347`. doi:10.48550/arXiv.1707.06347.

[159] Abida Sultana, Fahim Bashar, Mayukh Roy Chowdhury, and Aloizio Pereira Da Silva. A Software-Defined Radio based O-RAN Platform for xApp Conflict Detection and Mitigation. In *IEEE Military Communications Conference (MILCOM)*, pages 686–687, 2024. doi:10.1109/MILCOM61039.2024.10773739.

[160] Aloizio Da Silva, Mayukh Roy Chowdhury, Aditya Sathish, Asheesh Tripathi, Scott F. Midkiff, and Luiz A. Da Silva. CCI xG Testbed: An O-RAN Based Platform for Future Wireless Network Experimentation. *IEEE Communications Magazine*, 63(2):62–68, 2025. doi:10.1109/MCOM.001.2400322.

[161] *O-RAN gNB Overview: srsRAN Project gNB*. srsRAN Project, 2025. URL `https://docs.srsran.com/`. Accessed: 2025-11-26].

[162] srsRAN. srsRAN O-RAN SC RIC repository, 2025. URL `https://github.com/srsran/oran-sc-ric`. Accessed: 2025-11-26.

[163] Open5GS. Open5GS documentation: Open source implementation of 5G core and EPC, 2025. URL `https://open5gs.org/open5gs/`. Accessed: 2025-11-26.

Poznan University of Technology
Faculty of Computing and Telecommunications
Institute of Radiocommunications