POLITECNICO DI TORINO
Repository ISTITUZIONALE

XAI4C: XAI for Conflict Detection and Mitigation in O-RAN Near-RT RIC

Original
XAI4C: XAI for Conflict Detection and Mitigation in O-RAN Near-RT RIC / Varshney, Nancy; Puligheddu, Corrado; Badawy, Ahmed; Chiasserini, Carla Fabiana. - In: IEEE VEHICULAR TECHNOLOGY MAGAZINE. - ISSN 1556-6080. - (2025). [10.1109/MVT.2025.3625693]

Availability:
This version is available at: 11583/3004368 since: 2025-11-16T12:21:24Z

Publisher:
IEEE

Published
DOI:10.1109/MVT.2025.3625693

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

14 May 2026

XAI4C: XAI for Conflict Detection and Mitigation in O-RAN Near-RT RIC

Nancy Varshney, *Member, IEEE,* Corrado Puligheddu, *Member, IEEE,* Ahmed Badawy, *Member, IEEE,* Carla Fabiana Chiasserini, *Fellow, IEEE*

***Abstract*—The Open Radio Access Network (O-RAN) architecture introduces a modular approach to network design, enabling flexibility by decoupling hardware and software while fostering innovation in a multi-vendor ecosystem. However, its distributed framework creates challenges in managing network control conflicts across various components. In this work, we provide a comprehensive overview of conflict management in O-RAN focusing on Near-Real-Time RAN Intelligent Controller (Near-RT RIC) conflicts, and state-of-the-art strategies for addressing these challenges. We analyze the limitations of current approaches through the lens of Explainable AI (XAI). Thereafter, we propose XAI4C, a novel framework for conflict detection and mitigation among xApps in Near-RT RIC in O-RAN that leverages XAI to provide more transparent and explainable actions. Our solution improves detection accuracy by 30% and reduces the detection latency by 41% compared to a state-of-the-art benchmark, while also effectively mitigating conflicts to enhance network performance. Finally, we discuss future research directions and highlight key challenges in XAI-driven O-RAN conflict management, critical for adapting to the increasing complexity of next-generation network systems.**

***Index Terms*—Conflict management, explainable artificial intelligence (XAI), open radio access network (O-RAN), xApps.**

I. Introduction

The Open Radio Access Network (O-RAN) is a transformative approach that replaces traditional single-vendor Radio Access Networks (RANs) with flexible, multi-vendor architectures. O-RAN emphasizes openness, interoperability, and vendor diversity by leveraging open interfaces and standardized protocols. One prominent implementation of this idea is the O-RAN architecture, which facilitates integration between vendors, employs artificial intelligence (AI) and machine learning (ML) for intelligent decision-making, and features a disaggregated design that scales to meet network requirements. A key part of O-RAN is the use of rApps and xApps, which play distinct roles in network management. Specifically, rApps (Radio access network Applications) operate within the non-Real-Time RAN Intelligent Controller (Non-RT RIC) and are responsible for tasks such as long-term network optimization and policy-based management. In contrast, xApps function within the near-Real-Time (Near-RT) RIC on much shorter time frames (10 ms to 1 s), executing real-time tasks such as traffic steering, handover management, and interference management and anomaly detection.

O-RAN's decentralized architecture enables applications to operate as autonomous agents. Each agent is optimized for specific tasks, such as traffic management or interference control. However, these are often developed independently by third parties, and they may operate without explicit coordination. This modular, disaggregated design enhances agility and reduces reliance on single components, but also makes resource conflict management and system-wide optimization more challenging. Agents that rely primarily on localized or incomplete information may make suboptimal decisions that negatively affect network performance or intensify existing conflicts. Understanding agents' actions can be challenging, especially when their decision-making is powered by AI. While AI models are often viewed as "black boxes" due to their complex, data-driven behavior, even rule-based or statistical applications can be opaque, particularly when implemented as closed-source components. Coordination and conflict mitigation in O-RAN are generally handled by the underlying RIC platform or Service Management and Orchestration (SMO) [1].

While traditional rule-based conflict management techniques are computationally efficient, these are inherently rigid and poorly suited to the dynamic, multi-agent O-RAN environments. In contrast, existing AI/ML-based approaches, though more adaptive, typically function as black boxes and demand substantial training data while offering limited visibility into how their decisions impact network behavior.

In this work, we focus on a specific challenge in O-RAN – *real-time conflicts that emerge from the independent operation of third-party xApps within the Near-RT RIC.* Notably, these conflicts, which arise due to misaligned objectives or overlapping control actions, are fundamentally different from issues such as interface mismatches, protocol inconsistencies, or vendor-specific extensions, topics that lie beyond the scope of this study.

To address this problem, we propose XAI4C (Explainable AI for Conflict Management), a novel framework for near real-time conflict detection and mitigation in Near-RT RIC. Rather than depending on the internal logic of xApps, whether they are AI-based, rule-based, or statistical, XAI4C uses SHAP (SHapley Additive exPlanations) to correlate each xApp's inputs and outputs. This model-agnostic approach introduces explainability into conflict detection without requiring access to the xApp's internal design. As a result, XAI4C provides

Nancy Varshney and Corrado Puligheddu are affiliated with Politecnico di Torino, Torino, Italy (e-mail: nancy.varshney@polito.it; corrado.puligheddu@polito.it). Ahmed Badawy is affiliated with Qatar University, Doha, Qatar (e-mail: badawy@qu.edu.qa). Carla F. Chiasserini is affliated with Politecnico di Torino, Torino, Italy and CNIT, Parma, Italy, and with Chalmers University of Technology, Sweden (e-mail: carla.chiasserini@polito.it).

fine-grained, interpretable insights with modest computational overhead, offering greater transparency than rule-based conflict management methods and more real-time efficiency than traditional AI/ML solutions.

In the following, we first overview the O-RAN architecture, sources of conflict, and state-of-the-art research on conflict management in O-RAN (Sec. II), as well as the challenges of O-RAN conflict management through the lens of XAI (Sec. III). We then present an innovative solution to enhance conflict detection and mitigation in Near-RT RIC (Sec. IV), and its performance against a state-of-the-art scheme (Sec. V). Finally, we conclude the paper and envision future research challenges (Sec. VI).

II. O-RAN Conflicts and Their Management

This section discusses the O-RAN architecture, sources of conflict in O-RAN, and the existing conflict management frameworks.

A. O-RAN Architecture

An overview of a typical O-RAN architecture is shown in Fig. 1. It includes the Non-RT RIC and the Near-RT RIC. The Non-RT RIC, integrated within the SMO framework, is responsible for non-real-time functions (i.e., > 1 s latency), executed through rApps, leveraging AI/ML models, rule-based logic, heuristic algorithms, optimization frameworks, or/and data analytics to analyze historical and contextual data, producing policies, recommendations, and configurations. The SMO further oversees broader network functions such as policy enforcement, inventory management, RAN analytics, and AI/ML workflow services [2], [3]. The Near-RT RIC operates in the 10 ms–1 s latency range and hosts xApps that enforce these policies and perform closed-loop control based on real-time network conditions.

In O-RAN architecture, the term E2 Nodes refers to the Central Unit (CU) and Distributed Unit (DU), which are key components of the disaggregated RAN infrastructure. O-DUs handle real-time lower-layer processing, while CUs manage higher-layer functions. The Near-RT RIC communicates with these E2 Nodes over the E2 interface to gather data and apply control actions. DU is also managed by the SMO via the O1 interface for configuration, monitoring, and lifecycle management. The SMO also configures the Radio Unit (RU) through the Open Fronthaul (Open FH) plane [3]. The DU connects to the RU using the Open FH user and control planes to handle real-time data transmission and scheduling. Moreover, the Mobile Network Operators (MNOs) interact with both the SMO and the RICs through Human-Machine Interfaces (HMIs) to monitor, control, and debug xApps and rApps, as well as manage their deployment and behavior in the network.

B. Sources of Conflict

Though O-RAN offers flexibility, it introduces challenges not found in single-vendor systems where internal protocols streamline operations. Conflicts in O-RAN arise primarily due

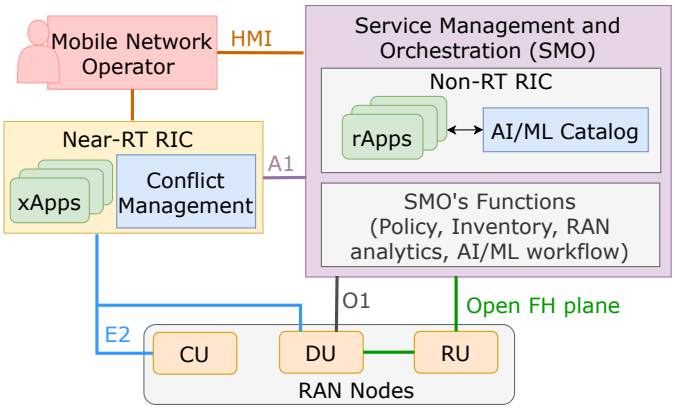

Fig. 1. Generalized O-RAN architecture.

to the decentralized and autonomous nature of applications, each with distinct objectives. E.g., in an urban scenario, authors in [1] found that running mobility load balancing and robustness optimization xApps without coordination may cause high radio link handover failures due to ping-pong effect.

Conflicts within the Near-RT RIC primarily arise among xApps that operate concurrently to manage various RAN functions. Since multiple xApps may attempt to modify the same control parameters simultaneously or adjust interdependent parameters without coordination, their actions can conflict and lead to suboptimal or even destabilizing outcomes. For instance, one xApp may change antenna tilt for better coverage while another alters cell offsets for improved handovers, resulting in counterproductive interference [1]. Additionally, xApps rely on shared data, and inconsistencies in how they interpret or act on these data further contribute to conflicts. Resource competition, e.g., on processing power, memory, or bandwidth, exacerbates these issues, leading to xApps underperformance and affecting overall network functionality.

Conflicts among xApps can manifest as: *(i)* direct conflicts, where multiple xApps simultaneously attempt to control the same resource or parameter; *(ii)* indirect conflicts, where independent actions affect interconnected KPIs; or *(iii)* implicit conflicts, which arise when xApps pursue different optimization goals, inadvertently degrading overall performance. These are considered horizontal conflicts, as they occur at the same time scale.

While the primary focus is on Near-RT RIC, conflicts can also emerge in the Non-RT RIC among rApps, which perform longer-term, policy-driven functions. These rApp conflicts typically stem from overlapping control scopes, divergent priorities (such as energy efficiency versus capacity), or inconsistencies in implementation, especially in multi-vendor ecosystems. Additionally, vertical conflicts may arise between rApps and xApps, due to the layered architecture and different operational time scales. An rApp might issue high-level policies via the A1 interface that aim to optimize long-term network behavior, such as minimizing energy consumption, while an xApp may focus on immediate goals like reducing latency. If the xApp fails to interpret or comply with the rApp's intent, or if the goals are inherently contradictory, it can result in behavioral inconsistencies and degraded network performance.

C. O-RAN Conflict Management Frameworks

The O-RAN technical report [3] defines a conflict management function in the Near-RT RIC to address conflicts among xApps (Fig. 1), though implementation details are left for future standardization.

The conflict management module follows three key processes: *(i)* detection, *(ii)* resolution, and *(iii)* avoidance [2]. Conflicts can be detected using several methods. KPI-based monitoring identifies issues by observing performance degradation, but it is slower to respond. Alternatively, analyzing control message exchanges between O-RAN components allows for faster, real-time detection, though it may introduce coordination complexity and overhead. Additionally, pre-deployment analysis such as examining control parameter subscriptions, can detect potential conflicts early, before xApps or rApps are deployed. Once conflicts are detected, resolution is achieved by reconfiguring settings, reallocating resources, or applying AI/ML-based optimizations. Conflict avoidance uses historical data and patterns to proactively prevent conflicts.

Existing approaches for conflict management can be broadly categorized into three main types: static, dynamic, and collaborative.

Static Conflict Management. Traditional methods rely on predefined rules, protocols, and policies tailored to resolve conflicts in predictable scenarios. They are relatively straightforward to implement and provide deterministic behavior with minimal computational and communication overhead. However, they struggle with dynamic conditions and unforeseen interactions due to limited adaptability. For example, hybrid priority-optimization schemes first satisfy high-priority goals, then optimize remaining objectives.

Dynamic Conflict Management. It overcomes the limitations of static methods by adapting to changing network conditions through model-based and rule-based decision making. Techniques like graph-based schemes and AI/ML-driven strategies enable nuanced conflict detection and resolution [4]. Similarly, model predictive control integrated with ML enhances multi-objective optimization in dynamic systems like self-organized networks, predicting network performance from historical data to resolve conflicts [5]. Dynamic strategies offer flexibility but face challenges like computational overhead, reliance on real-time data, and decision latency. Still, they are crucial in O-RAN, where traditional methods cannot meet the demands of dynamic environments.

Collaborative Conflict Management. This approach fosters cooperation and shared decision-making among xApps, as seen in team learning methods like Deep Q-Learning [6]. While effective in small-scale settings, it faces scalability, synchronization, and information-sharing challenges in larger or dynamic environments. A more advanced approach uses higher-level controllers for global objectives and lower-level agents for localized decisions, effective in scenarios modeled as Partially Observable Markov Decision Processes (POMDPs). E.g., [7] applies POMDPs to reduce service-level agreement (SLA) violations via dynamic resource allocation. Federated Learning (FL) further supports decentralized training with shared updates, while Vertical FL adds hierarchy for better responsiveness [8]. Though collaborative methods like team learning face issues like synchronization, latency, and scalability, they offer strong potential for balancing global objectives, local adaptation, and data privacy in O-RAN.

III. Challenges in Conflict Management

Despite the existing efforts, O-RAN conflict management still poses relevant challenges as set forth below.

Pre-action Conflict Detection Limitations. A relevant example is PACIFISTA [10], which analyzes xApps in a sandbox to create statistical profiles and predict conflict severity before deployment. However, the effectiveness of this type of mechanisms heavily depends on the quality and granularity of the available training data.

Post-action Conflict Detection Limitations. Implicit and indirect conflicts are particularly difficult to detect in post-action conflict detection due to the absence of direct relationships between parameters. Further, this challenge is compounded by latency, as conflicts can only be detected after the effects of xApp actions are observed. The framework in [1] uses real-time monitoring and analysis of control messages and performance management data within Near-RT RICs. However, it faces challenges in maintaining timely, accurate data for conflict resolution.

Scalability Issues. As application count grows, conflict management systems face scalability issues. For instance, the Deep Q-learning approach in [6] supports xApp cooperation but struggles to scale due to decision logic complexity. Scalable architectures and algorithms are needed to maintain efficiency in growing networks.

Computational Intensity of AI/ML-based Systems. Current conflict management systems, particularly AI/ML-based, can be computationally demanding. For example, the xApp distillation method in [4] lacks scalability due to high resource demands. Developing lightweight, energy-efficient algorithms and utilizing edge computing for localized conflict resolution can reduce resource usage. Also, reinforcement learning-based frameworks can improve system performance in dynamic environments.

Interpretability Issues. Regardless of whether xApps are AI-based, rule-based, or statistical, their decision-making logic is often opaque, especially when closed-source or developed independently by third parties. This lack of transparency complicates the detection of conflicting behaviors in the Near-RT RIC, where multiple xApps may act concurrently on shared control parameters. Explainable AI (XAI) helps address this challenge by analyzing how input features influence each xApp's output. Thus, it enables the identification of inconsistent or conflicting actions, without access to the xApp's internal logic. These insights can be used to flag potential conflicts or feed downstream classifiers for conflict detection. While XAI does not resolve conflicts directly, it can provide critical visibility that makes real-time, multi-agent conflict detection feasible. XAI is being applied to tasks like resource allocation and anomaly detection [13], but not yet to conflict management. For example, SliceOps [12] uses DRL with SHAP for slice optimization in the Non-RT RIC, focusing on resource management rather than resolving conflicts.

TABLE I
SURVEY OF CONFLICT MANAGEMENT STUDIES AND USE OF XAI IN O-RAN.

Reference	Focus Area	Methodology	Key Findings	Limitations
Adamczyk (2023) [9]	Challenges in Conflict Mitigation	Analytical study	Identified key challenges in conflict mitigation, including reliable conflict detection and efficient maintenance of mitigation configurations. Proposed pre-deployment conflict mitigation, real-time detection and resolution, and continuous supervision.	Limited observability of O-RAN components and lack of standardized testing methodologies.
Erdol et al. (2024) [4]	AI-based conflict mitigation	Knowledge distillation from multiple xApps	Proposed xApp distillation to combine multiple xApps into a single model, reducing conflicts and network outages.	Deployment of multiple xApps with overlapping functionalities leading to conflicts.
Adamczyk and Kliks (2023) [1]	Conflict mitigation framework in Near-RT RIC	Simulation-based analysis	Proposed a Conflict Mitigation Framework integrated into the Near-RT RIC to detect and resolve conflicts among xApps, demonstrating improved network performance.	Complexity in detecting and resolving various conflict types in real-time.
Brach del Prever et al. (2024) [10]	Conflict evaluation and management framework	Profiling pipeline and statistical modeling	Introduced PACIFISTA, a framework for detecting, characterizing, and mitigating conflicts among xApps. Provided sandbox testing environments and hierarchical graph analysis for pre-deployment conflict evaluation.	Ensuring compatibility among diverse xApps with potentially conflicting goals.
i14y Lab White Paper (2023) [11]	Conflict management strategies	Analytical study	Provided a comprehensive overview of challenges and solutions for conflict management in O-RAN, emphasizing standardization, collaboration, and centralized coordination through a Conflict Manager within Near-RT RIC.	Lack of standardized interfaces and procedures for conflict management.
Moysen and Giupponi (2018) [5]	Dynamic conflict management	Model Predictive Control (MPC) with ML	Proposed ML-enhanced MPC to address limitations of traditional models in complex, nonlinear systems. Improved multi-objective optimization in Self-Organized Networks (SON) by predicting network performance and resolving conflicts.	Struggles with highly variable or rapidly changing conditions in complex systems.
Qi et al. (2021) [8]	Collaborative conflict management	Federated learning with hierarchical control	Integrated Vertical FL for managing dynamic environments. Higher-level controllers managed global objectives, while lower-level agents adjusted decisions locally. Enhanced responsiveness in complex, partially observable scenarios.	Challenges in synchronization and communication among distributed agents.
Li et al. (2021) [6]	Team learning in O-RAN	Deep Q-Learning-based team learning	Proposed cooperative xApps to address conflicts in O-RAN. Demonstrated scalability in scenarios designed for explicit cooperation.	Limited scalability in scenarios with dynamically changing xApp cooperation requirements.
Dai et al. (2025) [7]	Conflict mitigation using POMDP	DRL-based radio resource management (RRM) Solution	Proposed a DRL-based RRM solution for RAN slicing with SLA guarantees, using a two-level scheduler (DRL-based inter-slice and proportional fairness-based intra-slice).	Challenges in implementing a two-level scheduler and dynamically managing resources to minimize SLA violations.
Rezazadeh et al. (2024) [12]	Explainable AI in resource allocation	DRL with SHAP-based XAI techniques	Proposed SliceOps framework to resolve resource allocation conflicts, ensuring interpretable decision-making and providing post-hoc explanations for actions.	Balancing interpretability with real-time responsiveness in high-demand scenarios.

Next, we propose and explain an XAI-based conflict management framework in the Near-RT RIC, designed to tackle the above challenges.

IV. THE XAI4C SOLUTION

We propose the XAI4C (XAI for Conflict Management), a KPI-based conflict management framework, for detecting and mitigating conflicts among xApps in Near-RT RIC. It leverages SHAP, an XAI technique that quantifies each feature's contribution to model outputs using Shapley values from cooperative game theory. Unlike traditional methods relying on statistical analysis or control message exchanges, XAI4C captures complex interactions among different features, supports real-time conflict detection, and enables continuous refinement of control policies. Furthermore, XAI4C addresses key challenges of the existing conflict management system in O-RAN, i.e.,

- *Interpretability Issues.* XAI4C addresses the "black-box" nature of xApps, whether AI-based, rule-based, or statistical, by providing detailed, actionable explanations based on input-output behavior. These interpretable insights support human-in-the-loop conflict resolution and enable a feedback loop, where recurring conflict patterns inform updates to detection logic and policy rules. While XAI4C does not autonomously resolve conflicts, it plays a key role in detecting them and guiding effective mitigation and adaptive management in complex O-RAN environments.
- *Computational Intensity of AI/ML-based Systems.* AI/ML-based methods for conflict detection often require frequent model retraining or repeated network simulations. In contrast, XAI4C adopts a more efficient two-step approach: it uses SHAP, a post-hoc explainability technique, to extract interpretable

features only when necessary (after a KPI violation), and then applies a lightweight classifier that is trained offline (discussed in detail in Sec. IV-A). This avoids retraining during runtime and reduces the computational burden significantly (as shown in Sec. V).

- *Scalability Issues.* XAI4C's modular and distributed architecture efficiently adapts to large-scale deployments. SHAP also leverages approximations, such as kernel SHAP, to balance accuracy and efficiency, and supports parallel computation, further enhancing its scalability.
- *Post-action Conflict Detection Limitations.* XAI4C enables granular analysis of how applications influence KPIs, identifying conflicts and ensuring quicker resolution in near real-time.

Next, we detail the proposed XAI4C architecture in Near-RT RIC.

A. The XAI4C Framework

XAI4C is seamlessly integrated into the O-RAN architecture, as depicted in Fig. 2. In the Near-RT RIC, the XAI-based conflict management system consists of three core components: KPI monitor (KPIMon), conflict detection module, and conflict mitigation module. The detection module integrates a SHAP engine and a classifier. The process starts with the deployment and activation of xApps by the App Manager (AppMgr) (Step 1). Once active, these xApps interact with RAN nodes via the E2 interface (Step 2) to execute control actions. Simultaneously, the KPI Monitor (KPIMon) continuously collects and tracks KPIs (Step 3).

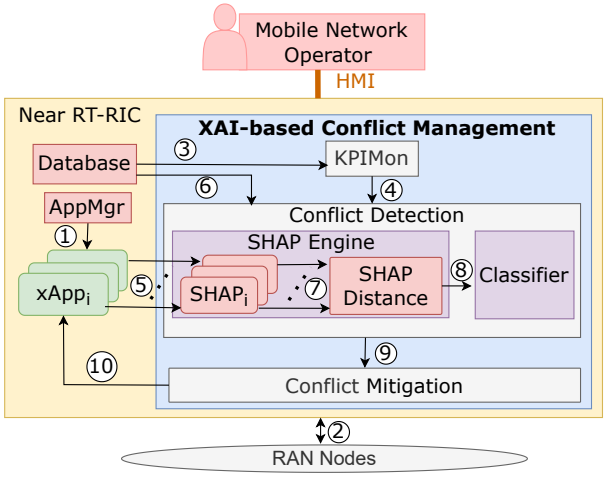

Fig. 2. XAI-based conflict management architecture.

When KPI degradation is observed, the conflict detection module is activated. In it, first the SHAP Engine (Step 4) computes SHAP values $x(s, f)$ for each active xApp's recent decisions to quantify the influence of each input feature $f \in \mathcal{F}$ on the KPI behavior under current network conditions $s \in \mathcal{S}$. Here, $\mathcal{S}$ is the set of network operational condition parameters (e.g., number of users, load, SNR) and $\mathcal{F}$ is the set of input features, encompassing both controllable configuration parameters (e.g., resource block (RB) allocation, modulation and coding scheme (MCS)) and monitored KPI values (e.g., latency, throughput). SHAP values are derived using a perturbation-based approach that alters one feature at a time while keeping others at baseline, attributing the resulting output deviation to that feature. Next, to determine whether the degradation results from internal conflicts, the conflict detection module (Step 6) retrieves operationally similar past conditions $\hat{s}$ from the database. Thereafter, for each feature $f \in \mathcal{F}$, it computes *SHAP distance* between the current network condition s and past network condition $\hat{s}$ (Step 7) as:

$$S_d = |x(s, f) - x(\hat{s}, f)| \,. \tag{1}$$

This distance quantifies shifts in feature influence, revealing whether xApp behavior under current conditions significantly deviates from baseline behavior. These distance vectors are fed into a trained classifier (Step 8) to distinguish between degradations due to external environmental changes and those caused by conflicting xApp behavior. Importantly, the classifier operates on SHAP distance rather than raw input data, which reduces both the dimensionality and the runtime computational complexity of the detection process.

If a conflict is detected, the conflict mitigation module (Step 9) is activated. It identifies the dominant contributing parameter (based on SHAP distance) and selectively adjusts the relevant controllable parameter of the responsible xApp(s) (Step 10) using one of these two strategies:

(i) Optimization-based mitigation, where the system searches the feasible range of the dominant parameter(s) (e.g., RB allocation, MCS) under system constraints to restore KPI performance while minimizing disruption to other xApps; or

(ii) ML-based mitigation, where a trained regression or policy model predicts the corrective adjustment based on historical recovery under similar conditions.

In both strategies, only the primary contributing parameter(s) are modified. The conflict is also reported to the SMO for managing policies on a long-term basis. These targeted adjustments are implemented in subsequent monitoring windows. This runtime process ensures that SHAP computations are not applied universally or constantly, but only in response to KPI violations. The SMO further coordinates with other network functions by sharing conflict information, thereby enabling joint optimization and consistent policy adjustments across services.

V. Case Study and Performance Evaluation

This section showcases conflict management among xApps in O-RAN using the proposed XAI4C framework.

The system model we consider (Fig. 3) integrates three primary xApps: Network Slicer, Energy Saver, and MCS Allocator. It foresees two network slices: enhanced Mobile Broadband (eMBB) for high data rates and ultra-Reliable Low Latency Communications (uRLLC). The Network Slicer employs a DRL mechanism with an actor-critic architecture to efficiently allocate resources according to slice-specific requirements. It optimizes eMBB throughput and uRLLC latency while operating within constraints of offered loads, CPU availability controlling the MCS, and RB budget. Trained on the Colosseum O-RAN dataset [14], the model leverages

a compressed state representation of key network metrics, including eMBB throughput, uRLLC latency, slice loads, RB budget, and maximum MCS, processed through an autoencoder for efficient feature extraction. Actions involve allocating RBs to slices to optimize their performance. The reward function balances maximizing eMBB throughput, minimizing uRLLC latency, and ensuring fair resource allocation to meet service-level objectives effectively. The MCS Allocator determines the maximum MCS index based on CPU availability and an RB budget capped at 50. Detailed relationships between CPU usage, RB budget, and MCS index are characterized in [15]. CPU availability for 50 RBs is set at 65%. For the maximum MCS, the Energy Saver predicts the RB budget required to support 1.2 times the total mean load over a monitoring window of length N time epochs.

Additionally, besides MCS Allocator and Energy Saver xApps, an upper control layer manages both the maximum MCS and the RB budget available to the Network Slicer. We consider that it maintains CPU availability at 70% for the maximum RB budget of 50 and determines the RB budget required to support 1.6 times the total mean load for the observation window.

Conflicts can arise when the Energy Saver or other xApps reduce resource availability, affecting network slice performance. The SHAP engine calculates SHAP values for the KPIs (eMBB throughput and uRLLC latency) and controllable parameters (MCS and RB budget) and the corresponding SHAP distances. The SHAP distances are fed to Logistic Regression-based classifier when KPI degradation is detected. If a conflict is detected, corrective mitigation actions are taken. We employ the optimization-based mitigation strategy described in Section IV-A. The system uses SHAP distance to determine which control parameters, MCS or RB budget, are most responsible for the observed degradation. If one feature shows a dominant SHAP distance, the system adjusts that parameter accordingly (e.g., increasing MCS or allocating more RBs). If both contribute significantly, it jointly tunes both parameters. These adjustments are reflected in the next monitoring window, ensuring real-time conflict resolution and maintaining network performance.

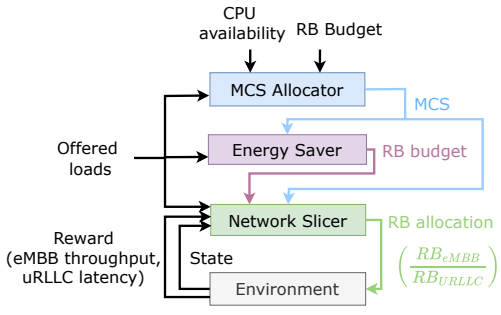

Fig. 3. System model of Energy Saver, MCS allocator, and Network Slicer interaction with the environment.

Numerical Analysis. We evaluate the system under three scenarios: s (stable network with only the Network Slicer active), $\tilde{s}$ (KPI degradation from conflicting xApps), and s' (external KPI degradation due to 20% increased SNR variability, with Energy Saver and MCS Allocator inactive). Slice load and average SNR are held constant for fair comparison. SHAP values are computed using the *shap.KernelExplainer*, which is trained on the same dataset used for the Network Slicer. We run the analysis for 200 time epochs using N=10, each of duration 1 ms, which results in a total of 20 monitoring windows.

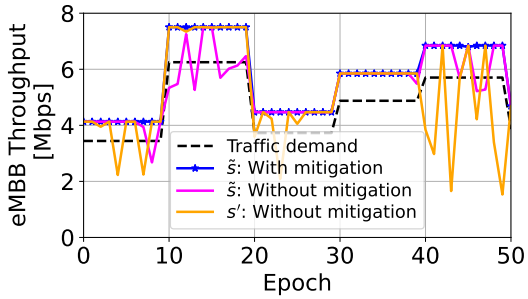

Fig. 4. Example of achievable eMBB throughput over 50 epochs in scenario $\tilde{s}$ (with mitigation), compared to scenario $\tilde{s}$ (without mitigation) and s' (without mitigation), along with the eMBB demand threshold.

To train the classifier, we define each input sample as a SHAP distance vector between a degraded scenario ($\tilde{s}$ or s') and the stable baseline s. The ground truth label is "Conflict" if the vector corresponds to $\tilde{s} - s$, and "SNR Deg" if it corresponds to $s' - s$. We use a Logistic Regression classifier trained on SHAP distances for eMBB throughput and uRLLC latency. Features are standardized, and SMOTE is applied to handle class imbalance. The data is split 70% for training and 30% for testing. Hyperparameters are tuned using GridSearchCV with 5-fold cross-validation across regularization strengths (C∈0.1, 1, 10), penalties ('l1', 'l2'), solvers ('liblinear', 'saga'), and iteration limits (500, 1000). The best model is used for real-time classification.

To illustrate the KPI temporal variation, Fig. 4 shows the achievable eMBB throughput over 50 epochs under three scenarios: $\tilde{s}$ (with mitigation), $\tilde{s}$ (without mitigation), and s' (without mitigation), compared to the eMBB traffic demand threshold. In scenario $\tilde{s}$, throughput remains consistently close to, or above, the demand threshold, demonstrating that the mitigation strategy effectively maintains eMBB performance despite potential internal conflicts. Conversely, scenario $\tilde{s}$ without mitigation and scenario s' exhibit frequent and sharp drops below the threshold, indicating degraded performance due to the absence of mitigation.

Fig. 5 presents the distribution of SHAP distances over 20 monitoring windows for key features, i.e., eMBB throughput, uRLLC latency, MCS, and RB budget, comparing the scenarios $\tilde{s}-s$ and $s'-s$. In the $\tilde{s}-s$ scenario, the limitations on the RB budget and maximum MCS result in larger variability and higher median impacts on all features, highlighting their collective effect on system performance under constrained resources. In the scenario $s'-s$, instead, they lead to smaller spreads and moderate median impacts, highlighting the more targeted effect of external environmental conditions. The classifier in the conflict detection module uses this information to

differentiate between the scenarios effectively.

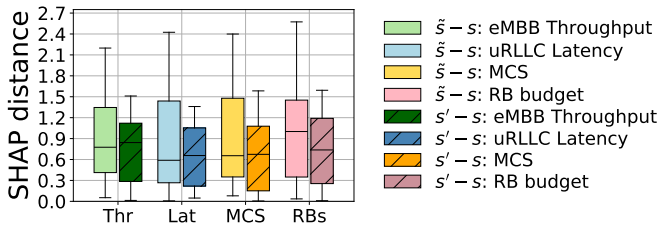

Fig. 5. Box plots of SHAP distances over 20 monitoring windows, comparing the conflict scenario $\tilde{s}$ and the external KPI degradation scenario s' with the baseline scenario s. SHAP features include eMBB throughput (Thr), uRLLC latency (Lat), MCS, and RB budget (RBs).

We compare XAI4C to a scheme inspired by the state-of-the-art PACIFISTA framework [10]. PACIFISTA predicts potential conflicts and their severity by analyzing statistical data from O-RAN xApps, using hierarchical graphs and historical KPI relationships. It employs the Kolmogorov-Smirnov (K-S) distance to quantify KPI deviations in conflict scenarios. PACIFISTA is primarily a pre-deployment tool, but its K-S method can be extended for online conflict detection. However, the approach inspired by PACIFISTA incurs substantial computational overhead due to its reliance on large datasets and complex statistical models, limiting scalability in dynamic, resource-constrained environments. For a fair comparison, both XAI4C and the approach inspired by PACIFISTA were evaluated using the same simulated O-RAN environment, including identical scenarios (s, $\tilde{s}$, s'), traffic loads, and KPI logging intervals. We measured detection accuracy by comparing predicted conflict labels to ground truth across 200 time epochs. Detection latency is defined as the number of monitoring epochs required to detect a conflict after KPI degradation begins. In our evaluation, we use fixed-size monitoring windows, each comprising 10 epochs. XAI4C successfully detects conflicts using just one such window (10 epochs), whereas its alternative requires at least 17 epochs of input for reliable detection. This corresponds to a 41% reduction in detection latency. Additionally, XAI4C achieves a 30% improvement in detection accuracy over the benchmark. It reflects XAI4C's ability to make faster detection decisions using fewer past observations, thereby improving responsiveness without increasing computational cost.

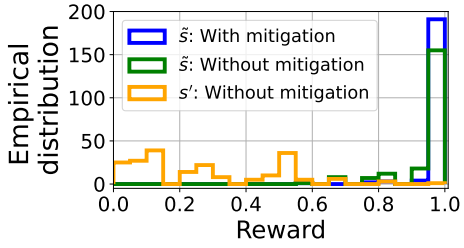

Fig. 6. Empirical distribution of average rewards computed over 200 monitoring windows of N=10 epochs, compared to scenario $\tilde{s}$ without mitigation, and in the scenario s' where no conflict mitigation is applied. Each value indicates the fraction of successful epochs (i.e., no KPI violations) within a window.

Fig. 6 shows the empirical distribution of average rewards, computed over a fixed monitoring window of N=10 epochs, for three scenarios: $\tilde{s}$ with mitigation, $\tilde{s}$ without mitigation, and s' (external degradation) without mitigation. In each epoch, a reward of 1 (0) is assigned for every satisfied (degraded) KPI. A KPI is considered degraded when the eMBB throughput falls below the eMBB traffic demand or the uRLLC latency exceeds the 0.1 s threshold. The total reward per epoch thus ranges from 0 (no KPI satisfied) to 2 (both satisfied). In scenario $\tilde{s}$, our mitigation scheme results in a significantly improved reward distribution, with a clear shift compared to scenario $\tilde{s}$. However, the reward does not reach 1 in all time epochs, as the conflict detection accuracy is limited to 61%, which constrains the ability to fully mitigate conflicts. For reference, the reward distribution for scenario s' is also provided, illustrating KPI degradation caused by external factors where conflict management is not applied.

Computational Cost of XAI4C. As mentioned, XAI4C avoids the overhead of repeated model retraining or network simulations by using a two-step approach. Our runtime evaluation shows that detecting conflicts using SHAP, for feature attribution followed by a lightweight, pre-trained classifier, requires only 0.33 s for 10 samples with 10 features and a classifier inference takes less than 1 ms. On the contrary, retraining a simple Logistic Regression-based ML model only over 10 iterations takes ∼3 s.

VI. Conclusion and Research Challenges

O-RAN is pivotal in facilitating AI-driven, dynamic network management but presents challenges in resolving conflicts among diverse applications. Our XAI4C framework tackles these challenges by utilizing XAI to provide transparent and model-agnostic analysis of xApp behavior. This enables fast, interpretable, and accurate detection and mitigation of conflicts in the Near-RT RIC, ultimately improving network reliability and efficiency. Compared to a state-of-the-art benchmark, XAI4C improves detection accuracy by 30%, with a 41% reduction in conflict detection latency. Additionally, XAI4C successfully mitigates conflicts, leading to better network performance. Its flexible design enables scalability across various network conditions, making it well-suited for dynamic environments. While this work focuses on conflicts in the Near-RT RIC, it lays the foundation for extending XAI4C to handle rApp conflicts in the Non-RT RIC and inter-layer conflicts, enabling unified, explainable conflict management across the RIC architecture.

However, integrating XAI into the complex O-RAN architecture presents several challenges.

- As the number of xApps increases, selecting the right SHAP variant is key for scalable model interpretation. TreeSHAP, suited for tree-based models like XGBoost, offers fast, exact explanations with constant runtime as feature count (number of xApps × KPIs) grows. In contrast, KernelSHAP, though model-agnostic and compatible with models like logistic regression and MLPs, incurs much higher computational cost due to its sampling-based nature. As shown in Fig. 7, KernelSHAP's runtime rises steeply, exceeding 3 s at 500 features for MLPs.

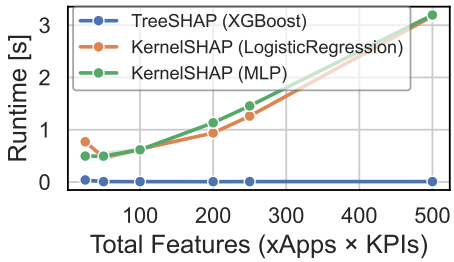

Fig. 7. Comparison of runtime with different SHAP variants as the total number of features increases.

- Maintaining an efficient feedback loop between conflict mitigation results and the AI/ML catalog is also key for continuous learning, without introducing unnecessary overhead.
- Finally, ensuring the accuracy of XAI outputs is critical, as noise in the outputs can affect conflict detection, making it important to establish reliable thresholds for conflict scores.

VII. Acknowledgment

Research reported in this publication was supported by the Qatar Research Development and Innovation Council ARG01-0525-230339. The content is solely the responsibility of the authors and does not necessarily represent the official views of Qatar Research Development and Innovation Council. N. Varshney and C. Puligheddu are supported by the European Union under the Italian NRRP of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 – program "RESTART").

Author Information

Nancy Varshney [M'23] is an Assistant Professor at BITS Pilani-Hyderabad Campus, India, since August 2025. Previously, she was an Assistant Professor at Politecnico di Torino, Italy since November 2023. She received her Ph.D. from IIT Delhi, India, in 2023. She was a Visiting Scholar at Aalto University, Finland, in 2022. Her research interests include 5G and beyond communications, edge computing, Open RAN, and beamforming.

Corrado Puligheddu [M'20] is an Assistant Professor at Politecnico di Torino, Italy, since 2023. He received his Ph.D. in Electrical, Electronics, and Communication Engineering from Politecnico di Torino in 2022. His research interests include 5G networks, edge computing, Open RAN, and machine learning, particularly in the context of demanding and innovative mobile applications.

Ahmed Badawy

Carla Fabiana Chiasserini [F'18] is a Professor with Politecnico di Torino, Italy. She worked as a visiting scholar and researcher at UCSD from 1998 till 2003. She was also a Visiting Professor at the Monash University, the Technical University in Berlin, and HPI Potsdam University. She is also an Affiliated Professor at Chalmers University of Technology, Sweden.

References

[1] C. Adamczyk and A. Kliks, "Conflict mitigation framework and conflict detection in O-RAN near-RT RIC," *IEEE Communications Magazine*, vol. 61, no. 12, pp. 199–205, 2023.

[2] O-RAN Alliance Work Group 3, "O-ran conflict mitigation technical report, v1.00," O-RAN Alliance, Technical Report WG-3 TR-CM v1.00, 2024.

[3] O-RAN Alliance Work Group 1, "O-ran architecture description," O-RAN Alliance e.V., Technical Specification O-RAN.WG1.TS.OAD-R004-v13.00, 2025.

[4] E. Erdol, A. Cicek, and A. Aydin, "xApp Distillation for conflict mitigation in O-RAN networks," *IEEE Trans. on Network and Service Management*, vol. 18, no. 1, pp. 12–23, 2024.

[5] J. Moysen, M. Garcia-Lozano, L. Giupponi, and S. 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*, vol. 13, no. 2, pp. 52–64, 2018.

[6] H. Zhang, H. Zhou, and M. Erol-Kantarci, "Team learning-based resource allocation for open radio access network (O-RAN)," in *IEEE ICC*, 2022.

[7] J. Dai, L. Li, R. Safavinejad, S. Mahboob, H. Chen, V. V. Ratnam, H. Wang, J. Zhang, and L. Liu, "O-RAN-Enabled Intelligent Network Slicing to Meet Service-Level Agreement (SLA)," *IEEE Transactions on Mobile Computing*, vol. 24, no. 2, pp. 890–906, 2025.

[8] J. Qi, Q. Zhou, L. Lei, and K. Zheng, "Federated reinforcement learning: Techniques, applications, and open challenges," *arXiv preprint arXiv:2108.11887*, 2021.

[9] P. Adamczyk, "Challenges in Conflict Mitigation for Open RAN Architectures," *Telecommun. Systems*, vol. 80, no. 4, pp. 517–532, 2023.

[10] L. Brach del Prever, M. Rossi, and S. Santucci, "PACIFISTA: A Framework for Conflict Evaluation in O-RAN xApps," in *Proceed. of the IEEE International Conference on Communications (ICC)*, 2024.

[11] i14y Lab, "Conflict Management Strategies in O-RAN: A White Paper," Online: https://i14y-lab.com/whitepapers/conflict-management.pdf, 2023, accessed: 2025-01-02.

[12] F. Rezazadeh, H. Chergui, L. Alonso, and C. Verikoukis, "SliceOps: Explainable MLOps for Streamlined Automation-Native 6G Networks," *IEEE Wireless Communications*, 2024.

[13] B. Brik, H. Chergui, L. Zanzi, F. Devoti, A. Ksentini, M. S. Siddiqui, X. Costa-Pèrez, and C. Verikoukis, "Explainable AI in 6G O-RAN: A Tutorial and Survey on Architecture, Use Cases, Challenges, and Future Research," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2024.

[14] "Colosseum ORAN dataset," https://github.com/wineslab/colosseum-oran-commag-dataset, 2024.

[15] S. Pramanik, A. Ksentini, and C. F. Chiasserini, "Cost-efficient slicing in virtual radio access networks," *Computer Communications*, vol. 209, 2023.