QACM: QoS-Aware xApp Conflict Mitigation in Open RAN

Abdul Wadud, *Graduate Student Member, IEEE*, Fatemeh Golpayegani, *Senior Member, IEEE*, and Nima Afraz, *Senior Member, IEEE*

***Abstract*—The advent of Open Radio Access Network (RAN) has revolutionized the field of RAN by introducing elements of native support of intelligence and openness into the next generation of mobile network infrastructure. Open RAN paves the way for standardized interfaces and enables the integration of network applications from diverse vendors, thereby enhancing network management flexibility. However, control decision conflicts occur when components from different vendors are deployed together. This article provides an overview of various types of conflicts that may occur in Open RAN, with a particular focus on intra-component conflict mitigation among Extended Applications (xApps) in the Near Real Time RAN Intelligent Controller (Near-RT-RIC). A QoS-Aware Conflict Mitigation (QACM) method is proposed that finds the optimal configuration of conflicting parameters while maximizing the number of xApps that have their Quality of Service (QoS) requirements met. We compare the performance of the proposed QACM method with two benchmark methods for priority and non-priority cases. The results indicate that our proposed method is the most effective in maintaining QoS requirements for conflicting xApps.**

***Index Terms*—Open RAN, conflict mitigation, QoS, xApp, Near-RT-RIC.**

I. Introduction

THE WIRELESS RAN has transformed over the past few decades as we have witnessed remarkable progress in wireless communication technologies. Many novel RAN concepts other than the Traditional Radio Access Network or Distributed Radio Access Network (D-RAN) have been introduced to support the more diverse and stringent network requirements of Fifth Generation (5G) and beyond communication systems including Cloud Radio Access Network (C-RAN), Virtualized Radio Access Network (V-RAN), Software Defined Radio Access Network (SD-RAN), and Open Radio Access Network. Open RAN is considered one of the most promising because of its disaggregated, virtualized, and vendor-neutral architecture. It provides a native framework to support Artificial Intelligence (AI)/Machine Learning (ML)-based control applications that enhance the network and resource management for the Mobile Network Operators (MNOs). Contrarily, the other versions of RAN architectures are typically deployed by a single vendor that works as a black-box solution, which risks vendor lock-in for the MNOs. Therefore, Open RAN is becoming more crucial for the MNOs.

The O-RAN ALLIANCE is a worldwide community of MNOs, vendors, and research & academic institutions that envisions the future as intelligent, open, virtualized, and fully interoperable Open RAN. Open RAN architecture has several significant challenges to overcome, as it is still less mature compared to other RAN architectures. Security & trust issues and interoperability are two of the major challenges. As the disaggregated Open RAN architecture uses multiple splits between components of the RAN protocol stacks, it opens many loopholes for attackers. A significant amount of studies are required to enhance trust and security in the Open RAN.

Interoperability remains another major challenge in Open RAN. When components from different vendors are assembled together, they should operate seamlessly, without any conflict or with minimal conflict. Conflicts negatively impact network performance and degrade the Key Performance Indicators (KPIs). There is neither a standardized conflict mitigation framework, nor is it defined how components from various vendors may coordinate, according to the O-RAN ALLIANCE. Therefore, in this study, we focus on post-action QoS-aware conflict mitigation within the RAN Intelligent Controller (RIC) to reduce the negative impact of conflicts. To elaborate, the RIC is responsible for network control within the open RAN architecture. It allows various vendors to deploy control applications aimed at specific network objectives, such as resource allocation, energy saving, mobility load balancing, and more [2]. Tasks that are not time-sensitive, taking longer than $1s$ to complete, are managed by Remote Applications (rApps) within the Non Real Time RAN Intelligent Controller (Non-RT-RIC). In contrast, tasks demanding completion in a time frame from $10ms$ to $\leq 1s$ are performed in xApps in the Near-RT-RIC. The xApps and rApps within the RIC oversee network operations and management. An example is an rApp that enhances network efficiency and minimizes delays in Vehicle to Everything (V2X) communication by optimizing the allocation of radio resources [3]. In a similar manner, an xApp can optimize QoS for a user group by efficiently

Manuscript received 15 January 2024; revised 2 June 2024; accepted 14 July 2024. Date of publication 24 July 2024; date of current version 27 August 2024. This work was supported in part by the European Union's Horizon Europe Research and Innovation Program through the Marie Skłodowska-Curie SE Grant under Agreement RE-ROUTE 101086343. This article was presented in part at IEEE CPSCom 2023 [1]. *(Corresponding author: Abdul Wadud.)*

Abdul Wadud is with the School of Computer Science, University College Dublin, Dublin 4, Ireland, and also with the Bangladesh Institute of Governance and Management, Dhaka 1207, Bangladesh (e-mail: abdul.wadud@ucdconnect.ie).

Fatemeh Golpayegani and Nima Afraz are with the School of Computer Science, University College Dublin, Dublin 4, Ireland.

Digital Object Identifier 10.1109/TGCN.2024.3431945

managing radio resources and dispatching targeted control signals to the RAN infrastructure [4]. There is a risk of adverse interactions affecting performance [5] given that these xApps and rApps are supplied by different vendors and operate on shared resources during network activities. Such interactions are called conflicts that must be identified and resolved to prevent significant declines in system performance.

The impact of conflict at the xApp level is significant. Recent studies [6], [7], [8] demonstrate, degradation in RAN KPIs when xApps operate independently without any conflict mitigation model. For instance, the authors in [6] examine Mobility Load Balancing (MLB) and Mobility Robustness Optimization (MRO) xApps, showing that a simple prioritization-based conflict resolution method can improve handover KPIs compared to scenarios without conflict mitigation. Similarly, [7] reveals that resource and power allocation xApps employing a team-learning based conflict mitigation method significantly outperforms independent operations in terms of throughput and packet drop rate (PDR). Finally, research by [8] shows that throughput drops by approximately 50% when Energy Saving (ES) and throughput maximization (TM) xApps operate together in a RAN slice due to conflicting configurations, reinforcing the need for research into xApp conflict mitigation in Open RAN.

The study in [6] focuses solely on conflict detection and mitigation frameworks, applying the xApp prioritization method exclusively to indirect conflicts. In contrast, [7] introduces a Deep Reinforcement Learning (DRL)-based team learning method that necessitates data sharing among xApps and incurs high computational costs, limiting scalability with an increasing number of xApps. Meanwhile, [8] presents a threshold-based mitigation approach, setting tolerance thresholds for each xApp to gauge conflict severity. If two xApps are producing conflict that is above the severity threshold, the MNO should not deploy them together to avoid conflict. However, it lacks discussion on how altering conflicting parameter values could mitigate this severity when it is above the threshold. These limitations motivate us to investigate a more efficient conflict mitigation method that is computationally less intensive, does not require data sharing among xApps, can effectively mitigate conflict severity in RAN, and can ensure the QoS of the network (see more in Section II-B).

In this article, we propose the QACM method to address various types of intra-component conflicts in the Near-RT-RIC while ensuring the individual QoS requirements of conflicting xApps. This proposed method extends our previous research [1], where we introduced two game-theory-based Conflict Mitigation Controllers (CMCs), namely Nash's Social Welfare Function (NSWF) and Eisenberg-Galle (EG) solutions. The NSWF is applied in non-priority scenarios where each xApp in conflict has equal preference, while EG is utilized in priority settings, allowing xApps to have varying preferences set by the MNO. However, these methods do not account for the QoS benchmarks of each associated KPI of the involved xApps during conflict mitigation. Hence, many xApps fall short of their QoS requirements with the provided solutions. Thus, this article proposes the QACM framework, considering the QoS benchmarks of conflicting xApps. We benchmark the NSWF and EG solutions against the proposed approach and compare their performances in Section VII.

The concept and architecture of the Conflict Mitigation System (CMS) are adopted from [6]. While our research mainly focuses on the CMC component of the CMS, the study in [6] concentrated on the Conflict Detection Controller (CDC). To the best of our knowledge the proposed QACM method is the first of its kind for RAN conflict mitigation.

We formulate the QACM method as an optimization problem and heuristic algorithm in Section VI and provide an in-depth discussion on the taxonomy of conflict in Open RAN in Section II. Furthermore, we present an example model with five stochastic xApps to illustrate different types of intra-component conflicts in the Near-RT-RIC and to theoretically analyze the performance of the proposed QACM method compared to benchmarks. The case study demonstrates that the proposed QACM method outperforms benchmarks in maintaining the QoS threshold of involved xApps. We conduct four different case studies that cover conflicting cases considering two or more involved xApps and methods for handling different types of conflicts both separately and together. This paper makes several significant contributions to the field of Open RAN conflict mitigation, particularly in the context of Near-RT-RIC:

- We provide a comprehensive taxonomy of conflicts in Open RAN and discuss the specific challenges and methodologies for mitigating intra-component conflicts within the Near-RT-RIC.
- we demonstrate different conflicts using an example model with five stochastic xApps.
- We provide a comprehensive overview of the CMS and its components for mitigating intra-component conflicts within the Near-RT-RIC.
- We propose the QACM method to specifically address various types of conflicts among xApps while ensuring the QoS requirements of each xApp are met.
- A novel optimization problem and heuristic algorithm for the QACM focusing on improving the computational efficiency and effectiveness of conflict mitigation.
- Four case studies alongside a simulation study of direct conflict in a RAN environment to empirically validate the effectiveness of the QACM method, showing its superiority in maintaining the QoS thresholds of involved xApps over traditional methods.
- Finally, the research offers insights highlighting the importance and practical implications of deploying xApps from diverse vendors in a standardized interface environment with a mitigation method.

These contributions collectively advance the state-of-the-art conflict mitigation strategies within the Open RAN ecosystem, offering scalable, efficient, and effective solutions for real-world deployment challenges. The remainder of the article is organized as follows: Section II covers the background of conflict with a state-of-the-art literature review. Section III discusses the system model for the proposed QACM method within the CMS framework. Section IV discusses the conflict management framework within the Near-RT-RIC architecture of Open RAN. Section V discusses the benchmark

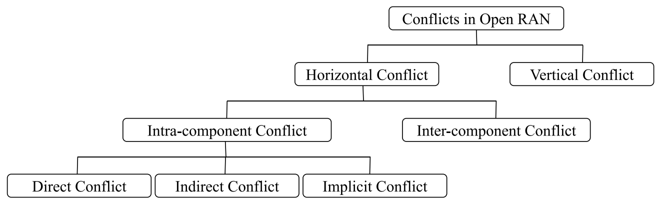


Fig. 1. Taxonomy of potential conflicts in Open RAN [1].

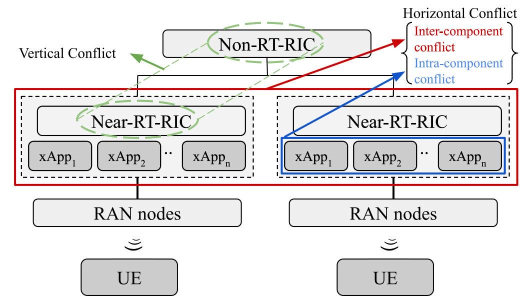


Fig. 2. Potential conflicting areas in Open RAN [1].

methods that are used to compare the performance with the proposed QACM method. Section VI elaborates on the proposed QACM method and its prerequisites. Section VII analyzes the performance comparison between the proposed and benchmark methods through case studies. Section VIII discusses the real-world application of the proposed method. Section IX addresses the limitations and future work. Finally, the paper concludes in Section X.

II. Background

A. Conflicts in Open RAN

In a conventional RAN setup with a single vendor, the vendor typically managed and resolved any conflicts within their own architecture. As the sole provider of the RAN system, they oversaw the design, setup, and fine-tuning of the network. They address any issues or incompatibilities within their exclusive ecosystem, leading to a more simple prevention and resolution strategy. However, the advent of Open RAN has changed this dynamic. This new network structure supports the incorporation of hardware and software from multiple vendors. While this enhances interoperability and adaptability, it also brings about possible discrepancies among the different components. Each vendor might employ distinct methods, enhancements, or settings, which can cause disputes when merging their technologies into the Open RAN environment and adversely affect the RAN‘s efficiency.

Consequently, these conflicts need to be identified and managed through appropriate network management [5].

1) Taxonomy of Conflicts in Open RAN: Recent studies [1], [6] have indicated that control decision conflicts in Open RAN architecture can manifest at various levels. These conflicts within Open RAN are typically divided into horizontal and vertical types, as depicted in Fig. 1.

A vertical conflict emerges between components at different layers of the Open RAN hierarchy. For instance, a conflict between a Near-RT-RIC and a Non-RT-RIC, as shown in Fig. 2, is identified as a vertical conflict. Conversely, a horizontal conflict arises among components at the same hierarchical level. An example is a dispute between two xApps within a single Near-RT-RIC or among adjacent Near-RT-RICs, which is classified as a horizontal conflict (refer to Fig. 2). Within a Near-RT-RIC, conflicts among xApps are termed intra-component conflicts, while those among xApps from neighboring Near-RT-RICs are called inter-component conflicts (see Fig. 2). Furthermore, intra-component conflicts can be broken down into direct, indirect, and implicit types. In this paper, we present a strategy to mitigate intra-component conflicts among xApps in the Near-RT-RIC. Within the Near-RT-RIC, autonomous xApps aiming for various optimization objectives can inadvertently create conflicting configurations by altering or affecting the same network parameter [5]. These are recognized as intra-component conflicts. Resolving direct, indirect, and implicit conflicts is challenging as the xApps involved are often developed and provided by different vendors and typically do not share information with each other [7].

Direct conflicts are easily identifiable by the internal conflict mitigation controller. A single xApps or a pair of xApps might request configurations that clash with the existing setup. Furthermore, several xApps might propose different values for the same parameter, leading to an evident direct conflict. The conflict mitigation controller evaluates these requests and determines which one should take precedence. This approach is referred to as pre-action resolution [5]. However, simply favoring one request over another is not always the best solution. A preferable strategy is to find an optimal configuration that reconciles the conflicting parameters, promoting fairness and aligning with the network's collective optimization objectives. In contrast, indirect and implicit conflicts are less obvious. An indirect conflict arises when one xApp‘s parameter adjustment inadvertently affects the operational area of another xApps. For instance, separate xApps controlling cell individual offset (CIO) and antenna tilts might influence the handover boundary. A change by an xApp managing remote electrical tilts (RET) or antenna tilts can, therefore, indirectly alter the performance of a CIO-focused xApps. Addressing this type of conflict involves post-action analysis to determine an optimal value for the contentious parameter [5]. Implicit conflicts occur when two xApps, each optimizing their respective targets, inadvertently degrade each other's performance. An xApp aimed at ensuring QoS for a set of users and another focused on minimizing handovers could, for instance, interfere with one another in subtle ways. Detecting and resolving this conflict is particularly challenging [5]. In this article, we introduce a conflict mitigation component designed to address all intra-component conflicts among xApps in the Near-RT-RIC.

B. Problem Background

The Near-RT-RIC is the core of control and optimization in Open RAN. xApps are the main components of the

Near-RT-RIC. According to the O-RAN Alliance's Open RAN architecture, it opens the network for smaller vendors to participate in the development of RAN components so that the over-dependency of MNOs on a handful of vendors can be alleviated. When these xApps from various vendors are deployed in the Near-RT-RIC, the possibility of having conflicting configurations among them is highly likely or certain as they share the same RAN resources [1], [6], [7], [9]. The common approach to deal with these conflicts is to develop a combined xApp that performs multiple objectives, for instance- traffic steering xApp [10], or enabling real-time data-sharing between these conflicting xApps for joint decision making using Multi Agent Reinforcement Learning (MARL) or other machine learning techniques [7]. From the MNO's perspective, the former may loop us back to the possibility of vendor-lock-in as it acts like a black-box of multiple tasks combined together. The latter will require real-time data sharing and management that adds excessive computational overheads in the latency sensitive Near-RT-RIC. Also, it fails when vendors are not interested in direct data sharing of their xApp's with others. In our previously published paper in [1], we proposed a game theoretic approach that can satisfy the low latency scenario considering the following few assumptions:

- Each xApp possesses the capability to independently learn, predict, and make optimal decisions based on the changing network state.
- xApps are capable of operating autonomously without intercommunication.
- Provided by various vendors, xApps function independently without forming groups or collective actions among themselves.
- All xApps utilize shared resources, leading to inherent conflicts of interest among them.

In our previous study [1], the NSWF is utilized to optimize a parameter that several xApps conflict over. This approach primarily aims to maximize the system's overall utility as shown in the equation (1). However, it does not consider the QoS requirements of individual xApps, potentially causing them to consistently fall below their QoS targets. To address this, we propose a QoS-aware conflict mitigation model considering all four aforementioned assumptions. This model focuses on ensuring that each xApp closely meets its individual QoS requirements as well as the overall network performance.

C. Related Research

Open RAN is a relatively new concept, and its conflict management aspects have not been thoroughly explored yet. However, there has been substantial research on conflict management within Self Organising Networks (SONs). The idea of xApps bears similarities to the concept of centralized SON Functions (SFs), as the RIC is seen as an evolution of the SON [11]. SON was introduced to simplify and automate the deployment and optimization of cellular RANs by removing the need for manual network element configuration. Consequently, it lowers the operational costs for mobile operators, enhances the Operating Expenditure (OpEx)-to-revenue ratio, and postpones unnecessary Capital Expenditure (CapEx). As the telecom industry moves towards open interfaces, virtualization, and software-centric networking, the SON ecosystem is gradually shifting from traditional Distributed SON (D-SON) and Centralised SON (C-SON) models to a framework based on open standards as xApps and rApps. This transition closely aligns with RAN programmability, fostering advanced automation and intelligent control through the RIC. The RIC, xApp, and rApp are equipped to support both near real-time D-SON and non-real-time C-SON functionalities, meeting the RAN automation and optimization requirements effectively [12]. Therefore, conflict mitigation in SON is relevant and essential to studying the conflict in Open RAN. Conflicts between MLB and MRO SFs are frequently studied within the context of SON [13], [14], [15]. The research in [13] addresses the conflict between MLB and MRO by restricting the Cell Individual Offset (CIO) parameter's range as determined by the MLB. The study conducted in [15] explored finding an optimal value for the CIO to enhance the mitigation of conflicts between these two SFs. Mu et al., in [14] introduced a coordination algorithm aimed at resolving the dispute between these SFs by adhering to pre-defined threshold values for Handover Ratio (HOR) and Call Block Ratio (CBR). A superior HOR above the threshold suggests improved performance for the MRO, while a lower CBR under the threshold signifies reduced cell-site congestion and thus better performance for the MLB. The study in [16] introduced a game-theoretic approach to mitigating conflicts among Cognitive Functions (CFs) in Cognitive Autonomous Networks (CANs). CANs represent an advanced version of SONs that employ machine learning and artificial intelligence to analyze network data and construct models depicting network behavior. The researchers developed a machine learning-based regression model for each CF using data gathered from the network. This data was collected for every CF through a simulated experimental setup that replicates real network conditions, encompassing all Input Control Parameters (ICPs) and KPIs associated with each CF. The NSWF was applied within three distinct conflict models to determine the optimal values for the conflicting ICPs while enhancing the overall utility of the network. This research was further expanded in [17] with actual network data obtained from a network simulator, considering both priority and non-priority scenarios for the CFs.

A recent study in [7] adopts a reinforcement learning based Deep Q-learning model for cooperative learning between power allocation and resource allocation xApps in Open RAN. The results showed higher throughput and lower packet drop rate while considering the team learning approach compared to the non-team learning approach. This approach demonstrated a new solution to resolve the conflicting problem that might be viable to adapt in Open RAN, but seems hard to excel for a higher number of xApps because of the complexity of joint decision-making for multiple participating xApps using the demonstrated framework. Moreover, xApps should be designed and developed with this proposed framework in mind, otherwise, the solution cannot be adopted. The research by [8] introduces a mitigation method based on setting tolerance thresholds for each xApp, which helps measure the

TABLE I
COMPARISON OF THIS STUDY WITH ARTICLES FOCUSING ON CONFLICT MITIGATION METHODS IN OPEN RAN

Article	**Method**	**Conflicts Covered**	**Ensures QoS**	**Parameter Control**	**Dynamic Mitigation**	**Data Sharing**	**Scalability**
Adamczyk et al. (2023) [6], [9]	xApp prioritization	Indirect	×	×	✓	×	Low
Zhang et al. (2022) [7]	Team learning	Indirect	×	×	✓	✓	Moderate
Prever et al. (2024) [8]	Severity threshold	Direct, indirect and implicit	×	×	×	×	High
This work	QACM method	Direct, indirect and implicit	✓	✓	✓	×	High

severity of conflicts. If the conflict between any two xApps exceeds this severity threshold, they should not be deployed simultaneously by the MNO to prevent further conflicts. However, this study does not explore how adjustments to conflicting parameter values could alleviate such severity once it surpasses the threshold. Table I provides a comparative summary of the proposed QACM method alongside the three state-of-the-art methods discussed.

Researchers have developed a conflict detection and resolution framework, as outlined in [6], which includes components for direct, indirect, and implicit conflict detection to effectively identify them within the Near-RT-RIC. The research primarily concentrates on the detection phase, while employing a simplistic priority-based system for conflict mitigation. Their findings demonstrated positive results, particularly with MLB and MRO xApps. Nonetheless, modifications to the current Open RAN architecture are necessary to integrate this solution. Reference [9] explores the potential challenges of mitigating various types of conflicts and discusses control loops for three different types of conflict mitigation approaches, including-preventive conflict mitigation, conflict detection and resolution, and supervision & adaptation. The preventive conflict mitigation approach suggests pre-deployment assessment of xApps and rApp in a digital-twin environment to detect potential conflicts and analyze their impacts on the network before deploying them to the actual RIC. The conflict detection and resolution approach is a post-action method in the live network that detects and resolves any type of conflict in near-real-time. The last of these three envisions to have a conflict supervisor component on top of the conflict detection and resolution framework that provides closed-loop monitoring and reconfiguration while mitigating conflict in the RIC.

In our earlier work presented in [1], which is a post-action conflict detection and mitigation framework, we primarily concentrated on the conflict mitigation component, assuming that the Conflict Detection (CD) component, with the support of the Performance Monitoring (PMon) component, accurately identifies conflicts. The conflict resolution strategy we adopted utilized two game-theoretic methods: NSWF and EG solution, for priority-based and non-priority-based scenarios, respectively. However, we observed that both NSWF and EG occasionally fail to guarantee satisfactory Quality of Service (QoS) for the maximum number of conflicting xApps. This inadequacy stems from not considering the QoS targets of individual xApp. We propose a QoS-aware conflict mitigation approach designed to ensure that the majority, if not all, xApps meet or exceed their specified QoS thresholds by identifying an optimal setting for the contentious ICP. We discuss this proposed method in Section VI.

III. SYSTEM MODEL

A. Assumptions and Notations

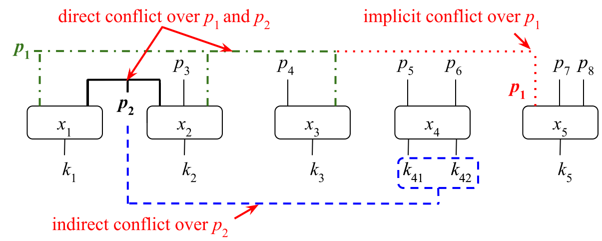
Let us assume that there are n xApps installed in the Near-RT-RIC. The set of xApps is denoted by X, where $X = \{x_1, x_2, \ldots, x_n\}$. Each xApp $x \in X$ has at least one associated KPI $k_j \in K$, where K indicates the set of all KPIs in the network. If there are multiple KPIs associated with a single xApp $x \in X$, we represent them as k_{ji}, where j denotes the index of the xApp, and i represents the index of the associated KPI. The KPIs of an xApp vary with the change of their associated input control parameters. We define the set of input control parameters as P, where $p \in P$. Each KPI has an individual QoS threshold $q_j \in Q$ to maintain. We define X' and Q' as sets of xApps with having a conflict over the conflicting parameter p_l and their associated KPIs, respectively. Since each KPI $k_j \in K$ may have a different unit, we convert them to a scalar unit by the function $U(p)$ and denote by $u \in U$. It is the utility function of $x \in X$ as a function of p. Converting the KPIs to the utility function is one of the most critical challenges of this proposed method, which is discussed in detail in Section VI-A. The objective of the proposed method is to select a value for the conflicting parameter $p_l \in P$ within a constrained range that minimizes the distance between the utility of xApp $x' \in X'$ and its QoS threshold $q' \in Q'$. s is the indicator of whether an xApp meets its QoS threshold or not for the estimated optimal value of the conflicting parameter p_l. w indicates the assigned weight to the conflicting xApps by the Conflict Supervision (CS) xApp. The functionalities of CS xApp are discussed in Section IV. The distance calculated between normalized QoS threshold and utility is very small, therefore, we use a constant

TABLE II
LIST OF NOTATIONS

Symbol	**Description**
Given Parameters:	
X	Set of all xApps in the Near-RT-RIC
P	Set of ICPs associated with all xApps in the Near-RT-RIC
K	Set of all KPIs associated with all xApps
$k \in K$	A particular KPI belongs to an xApp in the Near-RT-RIC
Q	Set of QoS thresholds for all xApps
$q \in Q$	A particular QoS threshold for an xApp
Q'	Set of QoS thresholds for all conflicting xApps
$q' \in Q'$	A particular QoS threshold for a conflicting xApp
X'	Set of all conflicting xApps
$x' \in X'$	A particular conflicting xApp
u	Utility of an xApp converted from KPIs
$U(p)$	Utility function to convert KPIs to utility u for a given value of the conflicting parameter p using z-score normalization
w	Priority weight assigned by the CS xApp
d	Variable representing the shortfall of xApp's KPI from its QoS threshold.
s	QoS indicator of xApps
δ	Binary variable to indicate KPIs to be minimized or maximized
ζ	Constant to balance the relative importance of the two parts of the objective function
Decision Variables:	
$p_l \in P$	Value of the conflicting parameter

ζ to tune the weighted distance. In our numerical analysis, we used $\zeta = 10^3$. All important notations used in this article are summarized in Table II.

Fig. 3 depicts the example model used for theoretically analyzing the efficacy of the proposed conflict mitigation method, where five stochastic xApps are considered. These xApps, installed in the Near-RT-RIC, are strategically modeled to encompass all intra-component conflicts. xApp x_1, x_2, and x_3 share the ICP p_1, thus exhibiting direct control decision conflict over p_1. Similarly, x_1 and x_2 have a direct conflict over p_2. We examine these two direct conflicts in three distinct scenarios: firstly, addressing direct conflict between two xApps; secondly, managing similar conflicts among more than two xApps; and lastly, simultaneously resolving two direct conflicts involving multiple xApps.



Parameter	Value	Parameter	Value
p_1	[-100, 100]	p_5	-20
p_2	[-10, 50]	p_6	[-50, 150]
p_3	-20	p_7	60
p_4	60	p_8	[-100, 150]

Fig. 3. Example model for direct, indirect and implicit conflict.

An indirect conflict is modeled considering p_2, p_5, and p_6, which belong to the same parameter group in the database. This means any change in their values impacts KPIs k_{41} and k_{42}. As defined in Section II-A, an indirect conflict arises when one xApp's parameter adjustment inadvertently influences the functional area of another xApp. This implies that a change in p_2, associated with x_1 and x_2, that inadvertently affects the KPIs associated with x_4, constitutes an indirect conflict of x_4 with both x_1 and x_2 over p_2. For instance- $P^G_{k_{41}} = \{p_2, p_5, p_6\}$ and $P^G_{k_{42}} = \{p_2, p_5, p_6\}$: here, $p_2 \notin I_{x_4}$, but it still affects k_{41} and k_{42}. Since $p_2 \in I_{x_1}$ and I_{x_2}, we can say there is an indirect conflict of these xApps with x_4 over p_2. Here, I_{x_1} and I_{x_2} are the set of ICPs for x_1 and x_2, respectively.

Lastly, an implicit conflict of x_5 with x_1, x_2, and x_3 over p_1 is modeled. This indicates that any alteration in p_1 inadvertently affects the KPI k_5 of x_5. Although p_1 is not directly linked as an ICP of x_5, it implicitly influences x_5's performance and acts as its implicit input, characterizing implicit conflicts. For instance- the parameter group for k_5 is $P^G_{k_5} = \{p_7, p_8\}$: This parameter group indicates that only modifying p_7 and p_8 should affect k_5. However, if during the RAN operation, k_5 gets affected by changing p_1, then we can say there is an implicit conflict between x_1, x_2, x_3 and x_5 over p_1 since $p_1 \in I_{x_1}$, I_{x_2}, and I_{x_3}. Including p_1 inside $P^G_{k_5}$ changes it to $P^G_{k_5} = \{p_1, p_7, p_8\}$, making it an indirect conflict. Thus, an implicit conflict is a vague form of indirect conflict that can only be detected during RAN operation but can be modeled as an indirect conflict once detected. Considering these conflicts and the values of all ICPs presented in the table in Fig. 3, we generate a conflict table for each xApp comprising all its associated ICPs and KPIs (see in Section VI-A and the Github repository at [18]). The values of the KPIs for different control inputs of the parameters are estimated using the Gaussian distribution equation as it often mirrors real-life scenarios [1], [16], [17]. Equations used for generating these KPIs are: $k_1 = 80 \times e^{-\frac{(p_1+0)^2}{2p_2^2}}$, $k_2 = 100 \times e^{-\frac{(p_1+p_3)^2}{2p_2^2}}$, $k_3 = 120 \times e^{-\frac{(p_1+45)^2}{2p_4^2}}$, $k_{41} = 120 \times$

$e^{-\frac{(p_6+(p_2-30))^2}{2p_5^2}}$, $k_{42} = 150 \times e^{-\frac{(p_6+(p_2-50))^2}{2p_5^2}}$, and $k_5 = -35 \times e^{-\frac{(p_8+(p_1-25))^2}{2p_7^2}}$. Afterwards, all xApps are trained with an Artificial Neural Network (ANN) regression model to enhance each xApp‘s KPI prediction capability for various settings of their ICPs using the generated dataset. Section VI-B provides a detailed discussion on the prediction aspect. In the numerical analysis, QoS thresholds used for each of the KPIs are: $q_1 = 55$, $q_2 = 95$, $q_3 = 85$, $q_{41} = 75$, $q_{42} = 80$, $q_5 = -25$.

IV. Conflict Management System Framework

In this article, we adopt the CMS framework as presented in [6] and [1]. This concept, incorporating a database, shared data layer, and messaging infrastructure, is congruent with the existing Near-RT-RIC architecture of the O-RAN ALLIANCE [5]. Our recent work [1] proposed a conflict management system consisting of three primary components: the Performance Monitoring (PMon), the CDC, and the CMC. These components, together with the database, are integral to detecting and mitigating intra-component conflicts within the Near-RT-RIC. The following sections discuss each of these components and the necessary database component, essential to the conflict management system framework.

1) Recently Changed Parameter (RCP): The Recently Changed Parameter (RCP) component within the database archives all parameters recently modified at the behest of various xApps. Each parameter is stored alongside its corresponding timestamp.

2) Parameter Group Definition (PGD): Within this database segment, parameters impacting the same network zone are cataloged. For instance, parameters like antenna tilts and cell individual offset, which both influence a cell’s handover boundary, are categorized together for each handover related KPI.

3) Recently Changed Parameter Group (RCPG): Changes to parameters within the Parameter Group Definition (PGD) are recorded in the Recently Changed Parameter Group (RCPG) section of the database. Here, each parameter alteration is logged with its timestamp and the associated parameters within the same group.

4) Parameter and KPI Ranges (PKR): The Parameter and KPI Ranges (PKR) database section compiles the minimum and maximum permissible values for each parameter, along with the relevant KPI for a specific cell.

5) Decision Correlated With KPI Degradation (DCKD): This part of the database is dedicated to recording individual KPI thresholds, which are determined based on the QoS and Service Level Agreement (SLA) requirements of the respective cells or networks.

6) KPI Degradation Occurrences (KDO): The KPI Degradation Occurrences (KDO) database component tracks occurrences of KPI degradation. This tracking is done subsequent to modifications in parameters by the xApps via RAN nodes and includes respective timestamps for each change.

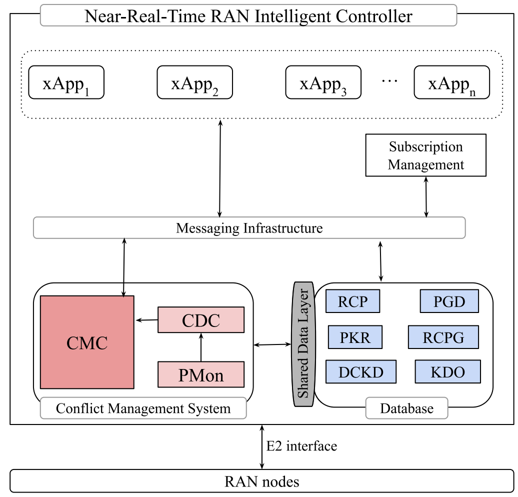


Fig. 4. The CMS Framework in the Near-RT-RIC.

The following subsections detail the functionalities of the core CMS components:

7) Performance Monitoring Component (PMon): The PMon in the CMS oversees monitoring and analysis of network KPIs. It gathers data from RAN nodes through the E2 interface, including network elements and user devices (refer to Fig. 4). This data, encompassing measurements and statistics, aids in KPI assessment against QoS thresholds. Deviations in KPI values, indicative of performance anomalies, are logged in the KDO database. Upon KPI breaches, PMon alerts the CDC to identify potential xApps conflicts.

8) Conflict Detection Controller (CDC): The CDC, as depicted in Fig. 4, detects various conflicts within the Near-RT-RIC, including direct, indirect, and implicit types, essential for a robust Open RAN architecture. Direct conflicts are identified through ICPs analysis during xApps deployment, while indirect and implicit conflicts are recognized via KPI degradation and parameter changes, respectively. The CDC then informs the CMC about any detected conflicts along with relevant details.

9) Conflict Mitigation Controller (CMC): The CMC addresses intra-component conflicts in the Near-RT-RIC by employing various conflict mitigation methods, including NSWF, EG or QACM. In case of detected conflicts, it suggests new parameter values that maximize or minimize certain objective functions among the involved xApps, based on their KPIs. The NSWF and EG discussed in Section V and QACM in Section VI are used to calculate this optimal value, considering both the most recent and the previous KPI values associated with the conflicting parameter.

In addition to these aforementioned components, we envision having a CS xApp deployed in the Near-RT-RIC that provides the MNO a solid control over the conflict mitigation system. The CS xApp closely monitors the network state and assigns weights to the conflicting xApps upon requests from the CMC. While assigning weights, it also considers the current policy configuration provided by the MNO. The

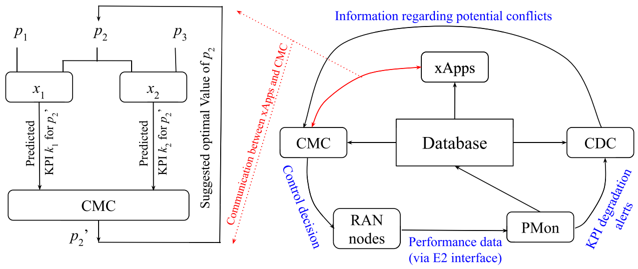


Fig. 5. Control Loop for the CMS.

MNO can update the policy configuration anytime that will immediately be effective to the CMS control-loop.

A. Control Loop for the Conflict Management System

The control loop of the CMS and its interconnected database components is depicted in Fig. 5. This loop illustrates the systematic flow and interaction between different components within the CMS highlighting the dynamic and responsive nature of the system. At the core of this loop is the constant monitoring and analysis of network performance, which is facilitated by the PMon component. This component continuously gathers and processes data from various RAN nodes, providing valuable insights into network performance through KPI assessment. Fig. 5 further demonstrates the sequential flow of operations, starting from data collection to conflict resolution. Upon detecting deviations in KPI values against predefined QoS thresholds, the PMon component triggers the CDC. The CDC then employs advanced algorithms to detect any potential conflicts between the xApps. Following conflict identification, the CMC takes charge, applying cooperative bargain game theory principles to resolve these conflicts, thereby ensuring optimal network performance. To facilitate this, the CMC engages in an iterative communication process with the involved xApps by exchanging information back and forth to calculate the optimal values of the conflicting parameters. This negotiation is visualized by the red line in the control loop depicted on the right side of Fig. 5. The CMC–xApps communication is illustrated on the left side of Fig. 5. The iterative bargaining process between the CMC and the involved xApps is crucial for reaching a consensus on the acceptable parameter values that satisfy the requirements of the network and the xApps without compromising on the overall system performance. The entire process embodies a cohesive and efficient approach to maintaining network integrity and performance in the Near-RT-RIC.

V. Benchmark

We consider NSWF and EG as the benchmark for performance comparison of the QACM method. The former is used when all conflicting xApps $x' \in X'$ have equal priority weights. The latter is used when different priority weights are assigned by the CS xApp based on the current network state and policy set by the MNO. Nash's equilibrium is well-known for computing maximum collective utility among players or agents in a multi-player or multi-agent scenario. It simply estimates the product of utilities of individual agents involved in the game. The Eq. (1) estimates the NSWF for all conflicting xApps $x'_i \in X'$ for the conflicting parameter p_l by iterative bargain within its optimal configuration range (see Section VI-C).

$$NSWF(p_l) = \Pi_{i\in[1,|X'|]} U_i(p_l), \forall i \in X' \tag{1}$$

The EG, on the contrary, is used when particular xApps require preferences over the other contentious xApps. For instance, let us consider a practical scenario stated in Section VIII where ES and MRO xApp have a direct conflict over Tx Power (TXP). If the current network state experiences high level of call drop rate, the CS xApp or the MNO can decide to put more preference to MRO over ES and help to rise values for TXP and CIO to increase throughput and broaden the handover boundary. In such a case, EG solution is essential, but NSWF fails as it doesn't consider priority cases. The following is the EG linear programming problem [1]:

$$\text{Maximize} \sum_{i\in[1,|X'|]} w_i U_i(p_l) \tag{2a}$$

$$\text{s.t.} \sum_{i\in[1,|X'|]} w_i = 1, \forall i \in X' \tag{2b}$$

$$p_l \geq p_l^{min,\text{opt}}, \forall l \in P \tag{2c}$$

$$p_l \leq p_l^{max,\text{opt}}, \forall l \in P \tag{2d}$$

$$\forall i \in \left[1, |X'|\right], |X'| \geq 2. \tag{2e}$$

The objective function (2a) represents the goal of maximizing the weighted utility of individual xApps within the set X' for a given parameter p_l. The utility function $U_i(p_l)$ captures the utility of xApp i for parameter p_l, which is influenced by specific network conditions and KPI metrics. The constraint (2b) ensures that the weights w_i assigned to each xApp sum to 1, thereby normalizing the influence of each xApp in the objective function. This normalization allows the aggregation of utilities to be proportionally representative of each xApp‘s importance. Constraints (2c) and (2d) define the permissible optimal range of the decision variable p_l. Finally, constraint (2e) ensures that the number of xApps involved in the optimization is at least two. It highlights the focus on scenarios where multiple xApps cooperate within the network.

VI. The Proposed Method

The proposed QACM method is designed to be deployed as the CMC component in the CMS, as shown in Fig. 4. In this article, we assume that all other components of the CMS framework perform their tasks efficiently, and the CMC is notified by the CDC when any direct, indirect, or implicit conflict occurs in the Near-RT-RIC. Afterwards, the CMC communicates back-and-forth with the conflicting xApps $x' \in X'$ to ensure that the maximum number of xApps meet their respective $q' \in Q'$ and to alleviate the negative impact of the transpired conflict. When an optimal value is obtained, achieving the objective goal of the proposed QACM method while satisfying certain constraints (see in SectionVI-D), the CMC forwards that optimal value of the conflicting ICP as a control decision to the respective xApps $x' \in X'$ and RAN

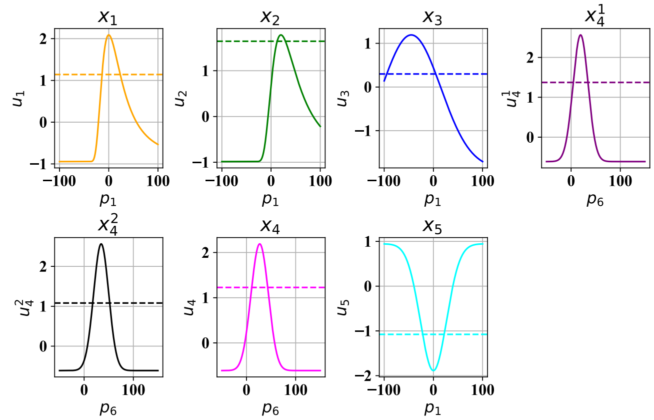


Fig. 6. normalized KPI values or utilities related to each xApp.

nodes. Certain prerequisites are necessary for the proposed QACM method to be deployed in the Near-RT-RIC, mainly the KPI prediction ability of individual xApp. This ability can also be developed as an independent KPI prediction xApp. These prerequisites are discussed in Sections VI-A and VI-B.

A. KPI to Utility Conversion

Converting KPIs to utilities and defining a common QoS threshold for each xApp in the presence of multiple KPIs associated with a single xApp poses a significant challenge to the proposed QACM method. We use z-score normalization [19] technique for converting the associated KPI of an xApp to utility. The reason for using z-score normalization technique over the min-max normalization technique (used in [1]) is primarily because it maintains the Gaussian distribution of the original KPI data, which is a crucial prerequisite for the proposed conflict mitigation model. Unlike min-max normalization, which simply re-scales the data to a fixed range, z-score retains the original distribution's properties, making it more suitable for utilities that rely on the underlying Gaussian-Normal characteristics.

To explain the conversion of KPIs to utility values, we consider an example model with five xApps and their respective ICP configurations as depicted in Fig. 3. The conflict tables for this model, including KPIs specific to each xApp and their corresponding QoS thresholds normalized using the z-score normalization technique, are available as datasets in our *QACM repository* [18].[1] Fig. 6 illustrates the normalized KPIs and their respective QoS thresholds. All horizontal dashed lines in each subplot indicate the normalized QoS thresholds $q \in Q$, and continuous curved lines refer to KPIs $k \in K$.

In our example model in Fig. 3, each xApp is linked to a single KPI with the exception of xApp x_4, which has multiple KPIs. This design choice reflects the complexity of real-world scenarios and demonstrates the necessity of converting KPIs into utility values. Because more number of KPIs associated with a single xApp adds extra bargain weights to the mitigation method. In a real-life scenario, for example, common xApps like Capacity and Coverage Optimization (CCO), ES, MRO, and MLB typically have multiple KPIs, as outlined in Tables IV and V. Therefore, having a single KPI representation for individual xApp is necessary to keep the computational latency of the mitigation model within the Near-RT-RIC‘s latency budget. Normalization of KPIs into utilities results in a uniform scalar measure, usually ranging between $[-3, +3]$. In a Gaussian distribution, approximately 99.7% of the data falls within three standard deviations from the mean. Therefore, after applying z-score normalization to the KPIs, which follow a Gaussian normal distribution, the resulting normalized scores predominantly range between -3 and $+3$. This standardisation allows for the integration of multiple KPIs associated with a single xApp to a single measure through various statistical techniques. However, combining these KPIs requires careful consideration, as some may be more significant than others. Also, combining different KPIs together might not always be meaningful. Addressing the QoS requirements of one KPI can sometimes meet the needs of other KPIs related to the same xApp. Thus, we emphasize the need for manual oversight in determining the relevant KPIs for conflict mitigation, given the complexity and contextual dependency of defining a KPI importance hierarchy for each xApp. This area, involving the identification of significant KPIs based on network conditions, presents substantial research opportunities, with potential applications for AI and ML methods which fall beyond the scope of the present work. To represent all associated KPIs as a single vector, techniques like Principal Component Analysis (PCA) or Factor Analysis can also be adopted. However, in this paper, we utilize a weighted-average method, which averages the KPIs with weights assigned by the MNO. The sixth figure from the top-left to the right in Fig. 6 presents the combined utility curve for the two KPIs of x_4, assuming equal importance for both. We presume that xApps are pre-trained with offline KPI prediction models capable of estimating KPI values based on provided ICPs. However, we train xApps with prediction models to evaluate the performance of the QACM framework and to demonstrate the prediction process, using the collected conflict table available at [18], based on the example xApp configuration shown in Fig. 3. This serves as a proof of concept for KPI prediction that we plan to investigate further in our future work. The subsequent section delves into the prediction of KPIs for xApps in greater detail.

B. KPI Prediction for xApps

KPI prediction aids the CMC–xApps interaction process by helping to estimate the optimal value of the conflicting parameter (as discussed in Section IV-A and illustrated in the left part of Fig. 5). We train two different regression models, ANN and Polynomial Regression (PR), for each xApp with individual conflict tables collected from the example model shown in Fig. 3. We use simplified models for KPI prediction as a proof of concept for the underlying principles of the QACM framework. More sophisticated models are required for KPI prediction in actual network environment [20]. We

[1] The dataset can be accessed directly via the following link: “QACM repository on GitHub”. For guidance on how to interpret these conflict tables, please refer to the README file within the repository.

TABLE III
PERFORMANCE COMPARISON BETWEEN ANN AND PR

xApps	PR			ANN		
	EVS	R-squared	MSE	EVS	R-squared	MSE
x_1	0.82	0.82	0.18	0.96	0.95	0.04
x_2	0.93	0.93	0.067	0.99	0.99	0.0006
x_3	0.99	0.99	0.0003	0.99	0.99	0.0045
x_4	0.96	0.96	0.02	0.98	0.98	0.01
x_5	0.81	0.81	0.185	0.98	0.98	0.015

discussed more about this in Section IX. The ANN comprises four hidden layers, each with 128 neurons. These layers utilize the hyperbolic tangent (*tanh*) activation function which is known for its efficacy in capturing non-linear relationships in the data. To mitigate the risk of overfitting, a dropout layer with a dropout rate of 0.2 is included after each hidden layer. This setup randomly sets a fraction of input units to 0 during training, helping to prevent complex co-adaptations on training data. The model terminates in an output layer with a single neuron employing a *linear* activation function that is ideal for regression tasks. For compiling the model, we used the *adam* optimiser that is a popular choice for deep learning applications, and the mean squared error loss function that is standard for regression problems. The model was trained over 10 epochs with a batch size of 10, balancing the efficiency and learning capability.

We compared the performance of ANN and PR models across five different xApps. The results as summarized in Table III indicate that the ANN model generally outperformed the PR model in terms of Explained Variance Score (EVS), R-squared, and Mean Squared Error (MSE). Specifically, the ANN model demonstrated higher EVS and R-squared values, suggesting it was more effective in capturing the variance and predicting the outcomes accurately for most xApps. Moreover, the ANN model exhibited lower MSE values, indicating that its predictions were closer to the actual values compared to those of the PR model. Therefore, we decide to use the ANN model for predicting KPIs of individual xApp.

C. Estimating the Optimal Configuration Range

When the CMC is informed about a conflict by the CDC, it immediately communicates with the set of involved xApps in X' and the CS xApp. The CS xApp quickly assesses the network policy configuration set by the MNO and the current network state, and subsequently, provides weights w_i for each involved xApp $x'_i \in X'$ so that $\sum_{i\in[1,|X'|]} w_i = 1$. Simultaneously, all the involved xApps $x' \in X'$ are asked to provide the individual optimal configuration range of the conflicting parameter p_l as $\{p_l^{min,x'_j}, p_l^{max,x'_j}\}$, where $p_l \in P$ and $x'_j \in X'$. Afterwards, the CMC estimates the overall optimal configuration range $\{p_l^{min,opt}, p_l^{max,opt}\}$ of the conflicting parameter p_l. Suppose, there are two xApps x'_1 and x'_2 involved in a particular conflict over parameter p_1. Upon request from the CMC, x'_1 and x'_2 send their individual optimal configuration range $\{p_1^{min,x'_1}, p_1^{max,x'_1}\}$ and $\{p_1^{min,x'_2}, p_1^{max,x'_2}\}$. The CMC estimates the optimal configuration range as $\{p_l^{min,opt}, p_l^{max,opt}\} \simeq \{min(p_1^{min,x'_1}, p_1^{min,x'_2}), max(p_1^{max,x'_1}, p_1^{max,x'_2})\}$.

D. QACM Method

The goal of the proposed QACM method is to minimize weighted distances of xApps KPIs from their respective QoS thresholds and a squared sum of QoS satisfaction indicators. This method estimates the optimal value of the conflicting parameter, which is later passed as a control decision to underlying RAN nodes by the CMC. Fig. 7 illustrates the flow diagram of conflict mitigation for the proposed method. We formulate the following optimization problem for QACM in this regard:

$$\text{Minimize} \quad \sum_{i\in[1,|X'|]} w_i d_i \times \zeta - \left(\sum_{i\in[1,|X'|]} s_i\right)^2 \tag{3a}$$

$$\text{s.t.} \quad \sum_{i\in[1,|X'|]} w_i = 1, \tag{3b}$$

$$\sum_{i\in[1,|X'|]} s_i \leq |X'|, \tag{3c}$$

$$d_i \geq 0 \quad \forall i \in \big[1, |X'|\big], \tag{3d}$$

$$d_i \geq \big(q'_i - U_i(p_l)\big) \times (1-\delta_i) + \big(U_i(p_l) - q'_i\big) \times \delta_i \quad \forall i \in \big[1, |X'|\big], \tag{3e}$$

$$U_i(p_l) \geq q'_i - M(1 - s_i) \quad \forall i \in \big[1, |X'|\big], \tag{3f}$$

$$s_i \in \{0, 1\} \quad \forall i \in \big[1, |X'|\big], \tag{3g}$$

$$\delta_i \in \{0, 1\} \quad \forall i \in \big[1, |X'|\big], \tag{3h}$$

$$p_l^{min,\text{opt}} \leq p_l \leq p_l^{max,\text{opt}}, \quad \forall p_l \in P, \tag{3i}$$

$$|X'| \geq 2. \tag{3j}$$

The Eq. (3a) represents the objective function of the optimization problem. It minimizes the weighted sum of distance reduced by the squared sum of satisfaction indicators s_i. It balances the need to keep xApps within desired QoS levels while maximizing overall satisfaction. The Eq. (3b) is the constraint that ensures the total sum of weights w_i assigned to each xApp is equal to 1. The total satisfaction score is constrained to the number of xApps by the Eq. (3c). Eq. (3d) and Eq. (3e) ensure that d_i is non-negative and measures the deviation from q'_i. Specifically, d_i measures the shortfall when $\delta_i = 0$ (KPI maximized) and the excess when $\delta_i = 1$ (KPI minimized). Eq. (3h) ensures δ_i is binary, with $\delta_i = 1$ for KPIs to be minimized and $\delta_i = 0$ for KPIs to be maximized. Binary condition in Eq. (3f) and Eq. (3g) ensure s_i accurately reflects compliance with QoS thresholds using a large constant *M* to enforce the binary condition. *M* is larger than the maximum possible value of $U_i(p_l)$. A bound is applied on the decision variable p_l in Eq. (3i). The permissible range for the decision variable is within the optimal configuration range discussed in Section VI-C. The final constraint in Eq. (3j) indicates that the number of xApps involved in this conflict mitigation approach is at least two because we consider scenarios involving at least two different xApps in a conflicting setting.

1) Complexity Analysis: In our optimization problem, the decision variable is primarily p_l, giving us only a unit variable. Additionally, we have w_i, s_i, and δ_i for each $i \in [1, |X'|]$, contributing twice to $|X'|$ variables. The d_i for each *i* also adds twice to $|X'|$ variables, leading to a total of $4|X'|$ variables.

The number of constraint analyses is equally straightforward. We have a single weighted sum constraint in Eq. (3b), a single satisfaction sum constraint in Eq. (3c), and $|X'|$ constraints each from the distance conditions in Eq. (3d) and Eq. (3e). Similarly, $|X'|$ constraints each from the satisfaction indicator condition in Eq. (3f), Eq. (3g) and Eq. (3h). Adding 2|*N*| constraints for the bounds on decision variables in Eq. (3i) considering |*N*| discrete values for p_l within the range $[p_l^{min,\text{opt}}, p_l^{max,\text{opt}}]$ and one for the minimum xApps involved in conflict in Eq. (3j), we reach a total of $4|X'| + 2|N| + 3$ constraints. This analysis highlights the computational complexity inherent to this latency-sensitive system.

E. QACM in a Dynamic Environment

For a dynamic scenario in the CMC, where the optimization problem is not tractable for large instance of conflicting xApps and optimal configuration bound of the conflicting parameter p_l, a heuristic approach can be adopted. By not tractable for large instances, we specifically refer to scenarios involving a large number of conflicting xApps and a wide range of optimal parameter configurations for p_l, which significantly increase the computational complexity. The complexity analysis in Section VI-D1 shows that exponential behavior can manifest as the size of $|X'|$ and |*N*| increases. Particularly, when each xApp interacts with the CMC during iterative bargaining, the number of interactions and the complexity of calculating the optimal value of p_l can grow exponentially. We developed Alg. 1 in light of the QACM optimization problem as follows.

In Alg. 1, the set of conflicting xApps X', the optimal bounds $\{p_l^{min,\text{opt}}, p_l^{max,\text{opt}}\}$ for the conflicting parameter p_l, weights assigned by the CS xApp w_i, and the QoS threshold q'_i for $\forall i \in [1, |X'|]$ are required as inputs. From steps 1 to 4, Alg. 1 initializes the required variables for estimating the *minCost* and the optimal value for p_l. The outer loop from steps 5 to 26 iterates within the range $\{p_l^{min,\text{opt}}, p_l^{max,\text{opt}}\}$, and the inner loop from steps 6 to 21 iterates for each conflicting xApp $x'_i \in X'$. The inner loop obtains $U_i(p_l)$ for an $x'_i \in X'$ in step 7, calculates the distance between q'_i and $U_i(p_l)$ based on the value of δ_i. If $\delta_i = 0$, the distance is measured as $q'_i - U_i(p_l)$ when $U_i(p_l) < q'_i$, otherwise it is set to 0. If $\delta_i = 1$, the distance is measured as $U_i(p_l) - q'_i$ when $U_i(p_l) > q'_i$, otherwise it is set to 0. The QoS indicator s_i is updated accordingly from steps 8 to 21, and estimates the weighted distance and stores it in the *cost* array in step 22. The algorithm breaks out from the inner loop in step 22. The final cost *fCost* of all xApps for a particular value of p_l within the range $\{p_l^{min,\text{opt}}, p_l^{max,\text{opt}}\}$ is calculated in step 23. If the calculated *fCost* is greater than *minCost*, the value of *minCost* and p_l^{opt} is updated using steps 24 to 26. When all iterations of the outer loop are completed, Alg. 1 returns the final p_l^{opt} in step 27.

Algorithm 1 Heuristic QACM for Dynamic Environment

Require: X', bound $\{p_l^{min,\text{opt}}, p_l^{max,\text{opt}}\}$ for p_l, w_i and q'_i for $\forall i \in [1, |X'|]$, ζ, δ
Ensure: Optimal value for p_l, p_l^{opt}.
1: Initialize a $cost$ array.
2: Initialize a p_l^{opt} variable.
3: Initialize $fCost = 0$, $minCost = \infty$ variables.
4: Initialize an array s as QoS indicator.
5: **for** each $p_l \in [p_l^{min,\text{opt}}, p_l^{max,\text{opt}}]$ **do**
6:   **for** each $i \in [1, |X'|]$ **do**
7:     Obtain predicted $U_i(p_l)$ from $x'_i \in X'$
8:     **if** $\delta_i = 0$ **then**
9:       **if** $U_i(p_l) < q'_i$ **then**
10:         Update $d_i = q'_i - U_i(p_l)$
11:         Update $s[i] = 0$
12:       **else**
13:         Update $d_i = 0$
14:         Update $s[i] = 1$
15:     **else**
16:       **if** $U_i(p_l) > q'_i$ **then**
17:         Update $d_i = U_i(p_l) - q'_i$
18:         Update $s[i] = 0$
19:       **else**
20:         Update $d_i = 0$
21:         Update $s[i] = 1$
22:     $cost[i] = w_i d_i \times \zeta$
23:   $fCost = \sum(cost) - (\sum(s))^2$
24:   **if** $minCost > fCost$ **then**
25:     Update $minCost = fCost$
26:     Update $p_l^{\text{opt}} = p_l$.
27: **return** p_l^{opt}.

1) Complexity Analysis: The complexity of Alg. 1 is primarily determined by its two nested loops. The outer loop iterates over each potential value of p_l within the range $[p_l^{min,\text{opt}}, p_l^{max,\text{opt}}]$. The inner loop, executed within each iteration of the outer loop, runs for each conflicting xApp in X', giving it a complexity of $O(|X'|)$. Each iteration involves a series of calculations, including obtaining predicted utilities, updating distances, and modifying QoS indicators and costs. Assuming the range of p_l consists of *N* discrete values, the overall complexity of the algorithm is approximately $O(N \cdot |X'|)$.

VII. Case Study

We consider four conflicting cases for analyzing the performance of the proposed QACM method as shown in Fig. 3. We use QACM Priority (QACMP) in the resulting figures only for comparing priority cases with the result of EG method. It is the same QACM method with different priority weights for involved xApps. In this section, the QACM illustrates results for a conflicting scenario where each of the conflicting xApp has equal weight, and it is compared with the result of NSWF method. All numerical analysis were conducted on a Python-based simulator same as [1]. The following critically analyzes the performance of

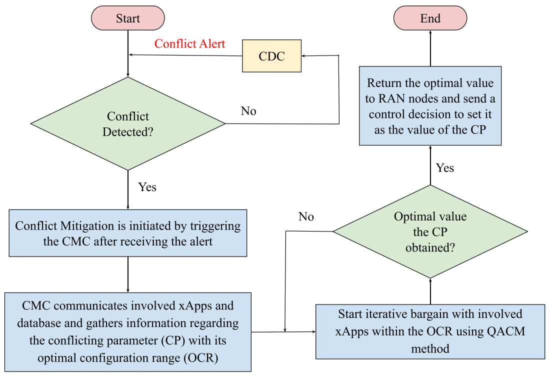


Fig. 7. Flow diagram of conflict mitigation with the proposed QACM method.

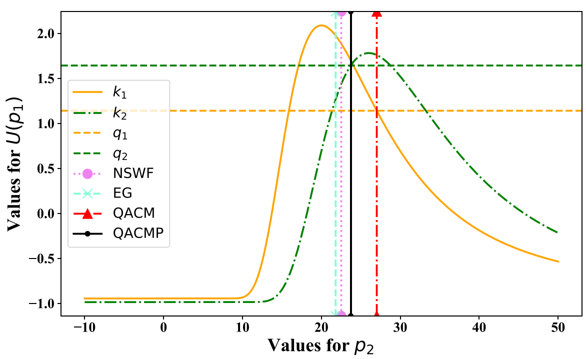


Fig. 8. Direct conflict between x_1 and x_2 over p_2.

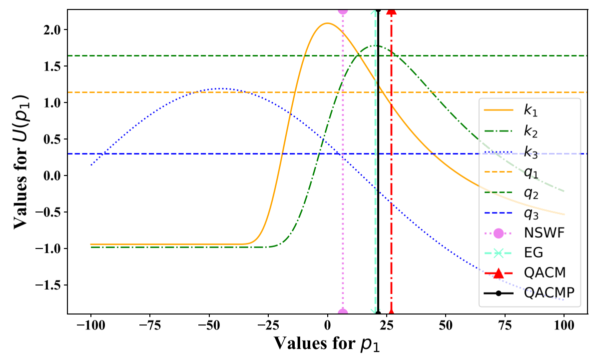


Fig. 9. Direct conflict among x_1, x_2 and x_3 over p_1.

the proposed model and compares its performance with two specific benchmarks as stated above.

A. Direct Conflict Between Two xApps

Fig. 8 illustrates a conflicting case between x_1 and x_2 over p_2. The curved lines, k_1 and k_2, indicate the utilities belonging to each xApp, and the dashed horizontal straight lines, q_1 and q_2, represent their respective QoS thresholds. The utility k_1 peaks at $p2 = 18$, leading x_1 to request the RIC to set $p_2 = 18$. Conversely, x_2 seeks to set $p_2 = 25$ to reach its maximum utility. To resolve this direct conflict, we execute the QACM, QACMP, NSWF, and EG methods. In a non-priority scenario with equal priority weights, $w_1 = 0.5$ and $w_2 = 0.5$, the QACM suggests setting $p_2 \approx 27$, where both xApps meet their individual QoS thresholds q_1 and q_2, respectively. In contrast, the NSWF suggests setting $p_2 \approx 23$, where only x_1 meets its requirement q_1, but x_2 fails to meet q_2. In a priority scenario, with $w_1 = 0.7$ and $w_2 = 0.3$, the QACMP suggests setting $p_2 \approx 25$, where both involved xApps meet their QoS thresholds. Conversely, the EG method, under the same priority configuration, suggests setting $p_2 \approx 22$, where only x_1 meets q_1. In both scenarios, the QACM ensures that the maximum number of involved xApps meet their individual QoS requirements.

B. Direct Conflict Among Multiple xApps

As depicted in Fig. 9, a conflict scenario involving three xApps over parameter p_1 is presented. In this case, the utility curves of the xApps, denoted as k_1, k_2, and k_3, are shown along with their respective QoS thresholds, represented by dashed horizontal lines q_1, q_2, and q_3. The utility curve k_1 for xApp x_1 reaches its maximum at a certain value of $p_1 = 0$, prompting x_1 to request the RIC to set $p_1 = 0$ to this optimal point. Similarly, xApps x_2 and x_3 each prefer different settings of p_1, as $p_1 = 20$ and $p_1 = -45$, respectively, to maximize their respective utilities. This scenario leads to a complex conflict among the three xApps, each vying for a different configuration of p_1.

To address this tripartite conflict, we apply the QACM method alongside the NSWF and EG solutions for both priority and non-priority settings. In a non-priority setting with equal weights for all xApps, the QACM method effectively finds a configuration of $p_1 \approx 23$ that satisfies the QoS requirements of x_1 and x_2. The NSWF methods, under similar conditions, suggest different configurations of $p_1 \approx -45$, favoring only x_3. In contrast, in a priority setting, the QACM method adapts to the assigned weights $\{w_1 = 0.1, w_2 = 0.2, w_3 = 0.7\}$ and finds an optimal configuration of $p_1 \approx 5$ that prioritizes and meets the QoS requirements of x_3 while meeting the same for x_1. At the same time, it keeps the deviation of k_2 from q_2 as smaller as possible. The EG method for the same setting finds an optimal configuration of $p_1 \approx 1$ that meets QoS requirements of both x_1 and x_3, but increases deviation between k_2 and q_2. This figure and its analysis underscore the efficacy of the QACM method in resolving conflicts involving multiple xApps with varying QoS requirements. It demonstrates the method's capability to navigate complex multi-party conflicts and identify configurations that balance the competing needs of different xApps in the Near-RT-RIC.

C. Concurrent Mitigation of Direct and Indirect Conflicts

Fig. 10 illustrates two distinct network states, η_1 and η_2. The former, η_1, refers to the network situation when a direct conflict over p_2 occurred between x_1 and x_2. The latter, η_2, indicates a later situation when the suggested value of p_2 by the QACM method to resolve the former conflict induces an indirect conflict with x_4. Within the context of Fig. 3, p_2 is implicated in a direct conflict between x_1 and x_2, as well as an indirect conflict involving x_4. In network state η_1, the

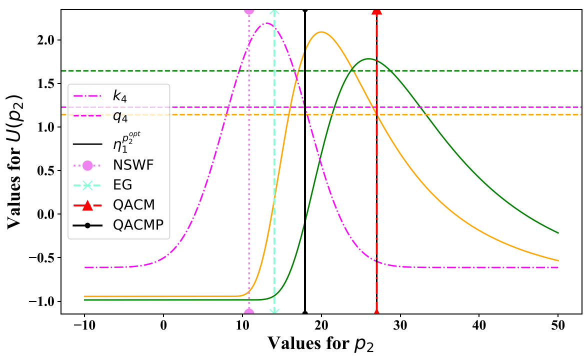


Fig. 10. Indirect conflict of x_4 with x_1 and x_2 over p_2.

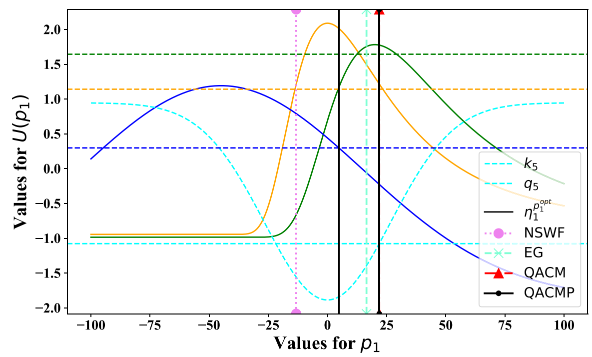


Fig. 11. Implicit Conflict of x_5 with x_1, x_2 and x_3 over p_1.

QACM non-priority method suggests a solution, $\eta_1^{p_2^{opt}} \approx 27$, to resolve the direct conflict, as depicted in Fig. 8. However, this solution inadvertently diminishes the utility of x_4 below its QoS threshold q_4, highlighting the indirect impact of p_2 on the KPIs of x_4. Consequently, x_4 advocates for setting p_2 to 15, where it achieves maximum utility, introducing a new conflict with x_1 and x_2.

The QACM method, along with the NSWF and EG solutions, is applied in both priority and non-priority settings. In a non-priority setting, the QACM and NSWF suggest $p_2 \approx 27$ and $p_2 \approx 11$, respectively, represented as η_2^{QACM} and η_2^{NSWF} in Fig. 10. The η_2^{QACM} maintains the previously suggested value, ensuring both x_1 and x_2 meet their respective QoS thresholds, due to the QACM method's inherent mechanism of maximizing the number of xApps that meet their individual QoS requirements.

Conversely, in a priority setting, the QACM method adjusts to the weights $\{w_1 = 0.1, w_2 = 0.2, w_4 = 0.7\}$ and identifies an optimal $p_2 \approx 18$ that satisfies q_1 and q_4, but not q_2. The EG method, with the same weights, proposes $p_2 \approx 14$, meeting only q_4. Thus, the QACM method surpasses both the NSWF and EG in non-priority and priority scenarios, respectively. This example showcases the intricate dynamics of conflict in Open RAN and emphasizes the QACM method's capacity to reconcile diverse QoS requirements among multiple xApps.

D. Concurrent Mitigation of Direct and Implicit Conflicts

Fig. 11 illustrates a scenario in which an implicit conflict, coupled with a direct conflict, emerges over the parameter p_1. We consider two network states, η_1 and η_2, akin to the previous case. Network state η_1 corresponds to the situation depicted in Fig. 9, where the QACMP scheme yields $p_1 \approx 5$ as the optimal solution for the conflict involving x_1, x_2, and x_3 over p_1. However, this resolution leads to a new conflict due to its detrimental effect on the utility of x_5. This conflict is presumed to be identified by the CDC, prompting the execution of the QACM, EG, and NSWF methods to determine an optimal value for p_1 once more, this time considering all four xApps. This is referred to as network state η_2.

In network state η_2, the QACM method and the NSWF provide solutions for p_1 in the non-priority case, with $\eta_2^{QACM} \approx 22$ and $\eta_2^{NSWF} \approx -13$, respectively. The QACM satisfies the QoS requirements for three xApps: x_1, x_2, and x_5. In contrast, NSWF fulfills the requirements for two xApps: x_1 and x_3. In the priority case, with the configuration $\{w_1 = 0.1, w_2 = 0.2, w_3 = 0.3, w_5 = 0.4\}$, the QACMP suggests the same value as QACM, while EG calculates $p_1 \approx 16$. Similar to the non-priority case, QACMP outperforms EG in meeting the QoS requirements of the involved xApps. This scenario underscores the complexities involved in simultaneously addressing direct and implicit conflicts.

VIII. Practical Application

This article primarily focuses on the theoretical development of a QoS-aware conflict mitigation approach. To illustrate the application of the proposed method in a real-life scenario, we consider four xApps, including CCO, ES, MRO, and MLB, as presented in Table IV and their respective objectives in Table V. From the list of ICPs in Table V, we identify four direct conflicts. Firstly, all four xApps share the TXP, leading to a direct conflict over TXP. Secondly, MLB and CCO have a direct conflict over Radio Electrical Tilt (RET). Lastly, MLB and MRO have two direct conflicts over CIO and Time-To-Trigger (TTT), respectively. Additionally, there is an indirect conflict between MLB and MRO xApps. All KPIs related to handover are influenced by this group of parameters: {TTT, CIO, TXP, RET, Handover Hysteresis (HYS)}. Any change in these parameters by MLB significantly affects the KPIs of MRO. For example, if MLB modifies RET, it directly impacts the handover boundary, potentially increasing the call drop rate. Direct and indirect conflicts between xApps are readily identifiable; however, establishing implicit conflicts among xApps is not feasible without live network simulation. Therefore, we are not considering any implicit conflicts among these four xApps in this section.

The aforementioned conflicts, including the implicit conflict, can be resolved using our proposed QACM method, as theoretically demonstrated in Section VII. However, practical validation requires these xApps to be deployed in a Near-RT-RIC and capable of predicting KPIs. We aim to validate the proposed QACM method for all conflicts and demonstrate the results in future works. However, we perform a simulation study considering a direct conflict between ES and CCO xApps over TXP in the following subsection considering an actual RAN scenario to showcase

TABLE IV
LIST OF XAPPS WITH THEIR RESPECTIVE KPIS

xApp	KPI	QoS Range	Unit	Minimum QoS Threshold			
				V2X	URLLC	mMTC	eMBB
CCO	SINR	-10 to 30 dB	dB [21]	20 dB	15 dB	10 dB	15 dB
	Downlink Throughput	10 Mbps to 1 Gbps	Mbps/Gbps [22]	500 Mbps	100 Mbps	10 Mbps	100 Mbps
ES	Energy Efficiency	0.1 to 10 (bit/Joule)	bit/Joule [23]	5 bit/Joule	2 bit/Joule	0.1-1 bit/Joule	1-3 bit/Joule
	Power Consumption	100 W to 1500 W	Watts [24]	1000 W	500 W	100 W	500 W
MRO	Handover Success Rate	90% to 99%	Percentage [14]	99%	98%	95%	97%
	Call Drop Rate	0% to 2%	Percentage [14]	$\leq$ 0.5%	$\leq$ 1%	$\leq$ 2%	$\leq$ 1.5%
	Call Block Rate	0% to 2%	Percentage [14]	$\leq$ 0.5%	$\leq$ 1%	$\leq$ 2%	$\leq$ 1.5%
MLB	Traffic Load	30% to 80%	Percentage	60%	50%	30%	40%
	Resource Utilization Rate	20% to 70%	Percentage	50%	40%	20%	30%

TABLE V
OBJECTIVES OF XAPPS BASED ON THEIR ICPS [17]

Name	ICPs	Objective
MLB	TTT, CIO, TXP, RET	Minimize Load, Maximize Resource Use
CCO	TXP, RET	Maximize Downlink Throughput, Minimize SINR
ES	TXP	Maximize Energy Efficiency, Minimize Power Use
MRO	TXP, TTT, CIO, NL, HYS	Maximize Handover Rate, Minimize Call Drops/Blocks

the performance of the proposed method. The simulation is performed without a RIC, and xApps are implemented as logic functions.

A. Validation Through Simulation

To validate the proposed QACM method, we used the MATLAB software for simulation and its 5G Toolbox with O-RAN 7.2 split [25]. The simulation involved two Next-generation NodeBs (gNBs) and ten User Equipments (UEs). The KPIs of interest, i.e., downlink throughput of CCO xApp and power consumption of ES xApp, were measured across different TXP values. The simulation parameters included a Reference Signal Received Power (RSRP) threshold of -110 dBm for handover, a frequency of 2.4 GHz, and a simulation time of 10 minutes with a time step of 100 ms. The QoS thresholds were set to 9.5 Gbps for throughput and 25 Wh for power consumption, ensuring that the system met the desired performance criteria. The input control parameters set during the simulations are: CIO of 2 dB, HYS of 0.5 dB, TTT of 0.1 ms, RET of 1.5 degree, and an adjustment interval of 1000 ms. The UEs are moving back-and-forth between the two gNBs with a randomly assigned velocity of 0 to 5m/s. Downlink throughput of CCO needs to maximize and power consumption of ES needs to minimize, therefore, we considered $\delta_{\text{CCO}} = 0$ and $\delta_{\text{ES}} = 1$ for the proposed QACM method.

A direct conflict occurs during the simulation when the ES xApp sets TXP to 6 dBm to minimize energy consumption in the network from its previous value of 40 dBm set by the CCO xApp. As a result, the downlink throughput KPI belonging to CCO, referred to as k_{CCO} in Fig. 12, experiences significant degradation below its threshold q_{CCO}. To mitigate

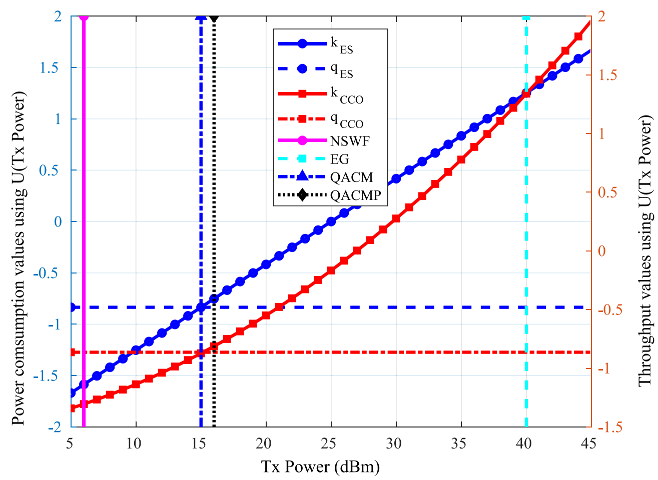


Fig. 12. Direct Conflict of ES and CCO xApps over TXP.

this conflict, we apply the QACM conflict mitigation method and compare its performance with benchmark methods, similar to the case studies discussed in Section VII. The optimal configuration range is set to 6 to 40 dBm for finding the potential optimal value of TXP. The QACM and QACMP methods, for non-priority and priority cases respectively, suggest an optimal TXP of 15 dBm and 16 dBm, whereas the NSWF and EG methods suggest 6 dBm and 40 dBm, respectively.

For non-priority cases with $w_{\text{ES}} = 0.5$ and $w_{\text{CCO}} = 0.5$, Fig. 12 demonstrates that the value suggested by NSWF for TXP meets the power consumption threshold q_{ES}, where k_{ES} is 142% below its threshold. However, the throughput k_{CCO} falls 39% below its threshold q_{CCO}. In contrast, QACM ensures that both xApps' KPIs are as close as possible to their respective thresholds, with k_{ES} exactly matching its threshold and k_{CCO}'s shortfall reduced from 39% to 3% relative to its threshold. For priority cases with $w_{\text{ES}} = 0.1$ and $w_{\text{CCO}} = 0.9$, the suggested value of TXP by EG results in power consumption increasing by 256% and throughput by 200% relative to their thresholds, creating an imbalance between the performances of these two xApps. Conversely, QACMP with a higher priority to the CCO xApp increases throughput by

3% while only increasing power consumption by 5%, ensuring that both xApps either meet their thresholds or stay as close as possible to them. Therefore, we can confirm that the proposed QACM method outperforms other benchmark methods in QoS-aware conflict mitigation and maximizes the number of xApps meeting or closely approaching their respective QoS thresholds.

IX. Limitation and Future Works

As discussed in Section VIII, the primary focus of this work is the theoretical foundation of the QACM method. While the KPI prediction for xApps plays a crucial role in our proposed approach, the current study employs a simplified ANN model to illustrate the framework's potential. It is important to note that in a real RAN environment, KPI prediction is a complex process influenced not only by ICPs but also by the dynamic state of the network. The simplified KPI prediction model used herein serves as a proof of concept for the underlying principles of the QACM framework. Recognizing the need for a more comprehensive approach to KPI prediction in actual network scenarios, we plan to conduct an in-depth investigation into this aspect in future research endeavors. Recent development [20] on KPI prediction in RAN shows the way to reach our vision.

X. Conclusion

In this article, we proposed the QACM method that mitigates intra-component conflicts within the Near-RT-RIC of Open RAN architectures. By integrating elements of cooperative game theory, particularly NSWF and EG solutions, the QACM method effectively balances conflicting parameters while upholding the individual QoS requirements of xApps. This approach not only enhances the flexibility and efficiency of network management but also offers a standardized framework for conflict resolution among diverse network applications. The comparative analysis with benchmark methods in priority and non-priority scenarios further establishes the QACM method's superiority in maintaining QoS thresholds under conflicting conditions. However, it is crucial to recognize the simplicity of the KPI prediction model used in this research may not comprehensively represent the complexity and dynamism of real-world RAN environments. This limitation underscores the need for more comprehensive KPI prediction models and the integration of the conflict mitigation framework introduced in this paper within the testbed environments that we aim to investigate in future. Also, we envisioned the concept of CS xApp for conflict supervision which is an integral part of the QACM based conflict mitigation system. We aim to dive deeper into the implementation of each of these components. Our future research will focus on bridging the gap between theoretical models and practical applications. In doing so, we aim to reinforce the role of QACM as an essential component in Open RAN, ensuring optimized performance and enhanced user experience.

References

[1] A. Wadud, F. Golpayegani, and N. Afraz, "Conflict management in the near-RT-RIC of open RAN: A game theoretic approach," 2023, *arXiv:2311.13389*.
[2] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," 2022, *arXiv:2202.01032*.
[3] M. Dryjański. "Traffic management for V2X use cases in O-RAN," Accessed: Nov. 9, 2023. [Online]. Available: https://rimedolabs.com/blog/traffic-management-for-v2x-use-cases-in-o-ran/
[4] *O-RAN Use Cases Analysis Report, Version 09.00*, O-RAN Alliance e. V., Alfter, Germany, Oct. 2022.
[5] "O-RAN working group 3 (near-real-time RAN intelligent controller and E2 interface Workgroup), near-RT RIC architecture," O-RAN Alliance e. V., Alfter, Germany, document O-RAN.WG3.RICARCH-R003-v04.00, March 2023.
[6] C. Adamczyk and A. Kliks, "Conflict mitigation framework and conflict detection in O-RAN near-RT RIC," *IEEE Commun. Mag.*, vol. 61, no. 12, pp. 199–205, Dec. 2023.
[7] H. Zhang, H. Zhou, and M. Erol-Kantarci, "Team learning-based resource allocation for open radio access network (O-RAN)," in *Proc. IEEE Int. Conf. Commun.*, 2022, pp. 4938–4943.
[8] P. B. del Prever et al., "PACIFISTA: Conflict evaluation and management in open RAN," 2024, *arXiv:2405.04395*.
[9] C. Adamczyk, "Challenges for conflict mitigation in O-RAN's RAN intelligent controllers," in *Proc. Int. Conf. Softw., Telecommun. Comput. Netw. (SoftCOM)*, 2023, pp. 1–6.
[10] M. Dryjański, Ł. Kułacz, and A. Kliks, "Toward modular and flexible open RAN implementations in 6g networks: Traffic steering use case and O-RAN xApps," *Sensors*, vol. 21, no. 24, p. 8173, 2021.
[11] (Mavenir, Richardson, TX, USA). *RIC as the Next Generation SON for Open RAN and More*. (2023). [Online]. Available: https://www.mavenir.com/resources/ric-as-the-next-generation-son-for-open-ran-and-more/
[12] (SNS Worldwide Ltd., Dubai, UAE). *SON (Self-Organizing Networks) in the 5G & Open RAN Era, pp. 2022–2030—Opportunities, Challenges, Strategies & Forecasts*. Accessed: Jan. 1, 2024. [Online]. Available: https://www.snstelecom.com/son
[13] Z. Liu, P. Hong, K. Xue, and M. Peng, "Conflict avoidance between mobility robustness optimization and mobility load balancing," in *Proc. IEEE Glob. Telecommun. Conf.*, 2010, pp. 1–5.
[14] P. Mu, R. Barco, and S. Fortes, "Conflict resolution between load balancing and handover optimization in LTE networks," *IEEE Commun. Lett.*, vol. 18, no. 10, pp. 1795–1798, Oct. 2014.
[15] M. Huang and J. Chen, "A conflict avoidance scheme between mobility load balancing and mobility robustness optimization in self-organizing networks," *Wireless Netw.*, vol. 24, pp. 271–281, Jan. 2018.
[16] A. Banerjee, S. S. Mwanje, and G. Carle, "Game theoretic conflict resolution mechanism for cognitive autonomous networks," in *Proc. Int. Symp. Perform. Eval. Comput. Telecommun. Syst. (SPECTS)*, 2020, pp. 1–8.
[17] A. Banerjee, S. S. Mwanje, and G. Carle, "Toward control and coordination in cognitive autonomous networks," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 1, pp. 49–60, Mar. 2022.
[18] "QACM repository," Jan. 15, 2024. [Online]. Available: https://github.com/dewanwadud1/QACM
[19] S. D. Colan, "The why and how of Z scores," *J. Am. Soc. Echocardiogr.*, vol. 26, no. 1, pp. 38–40, 2013.
[20] N. P. Tran, O. Delgado, B. Jaumard, and F. Bishay, "ML KPI prediction in 5G and B5G networks," in *Proc. Joint Eur. Conf. Netw. Commun. 6G Summit (EuCNC/6G Summit)*, 2023, pp. 502–507.
[21] A. Alhammadi, W. H. Hassan, A. A. El-Saleh, I. Shayea, H. Mohamad, and W. K. Saad, "Intelligent coordinated self-optimizing handover scheme for 4G/5G heterogeneous networks," *ICT Exp.*, vol. 9, no. 2, pp. 276–281, 2023.
[22] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan, "Beyond throughput, the next generation: A 5G dataset with channel and context metrics," in *Proc. 11th ACM multimedia Syst. Conf.*, 2020, pp. 303–308.
[23] J. Wu, R. Tan, and M. Wang, "Energy-efficient multipath TCP for quality-guaranteed video over heterogeneous wireless networks," *IEEE Trans. Multimedia*, vol. 21, no. 6, pp. 1593–1608, Jun. 2019.
[24] O. Shurdi, L. Ruci, A. Biberaj, and G. Mesi, "5G energy efficiency overview," *Eur. Sci. J.*, vol. 17, no. 3, pp. 315–27, 2021.
[25] M. Arafat Habib et al., "Transformer-based wireless traffic prediction and network optimization in O-RAN," 2024, *arXiv:2403.10808*.

Abdul Wadud (Graduate Student Member, IEEE) received the M.Sc. degree in computer science from South Asian University (SAU), India, in July 2020. He is a Ph.D. Researcher with the School of Computer Science, University College Dublin, Ireland, and a Research Associate with the Bangladesh Institute of Governance and Management, Dhaka. He has published research articles in top-tier journals like IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, and IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. His research interests include open RAN, wireless mobile networks, optical networks, optimization, and AI/ML for communication. He was awarded the Prestigious President Scholarship and the SAU Special Scholarship at SAU.

Fatemeh Golpayegani (Senior Member, IEEE) received the Ph.D. degree from Trinity College Dublin in 2018. She is an Assistant Professor with the School of Computer Science, University College Dublin (UCD). She joined UCD in 2019. She leads the multiagent and sustainable solutions research group, including three postdoctoral researchers, and six Ph.D. students. Within her research group the main focus is on developing AI-powered decision making algorithms, and optimization algorithms for complex systems and environments. Her research is applied in several fields, including intelligent transport systems, power systems, and bio systems. She has secured over two million euros in funding through international and national sources. Specifically, she is coordinating an MSCA SE Project, RE-ROUTE funded under Horizon Europe. She is the co-PI of another EU Project, augmented CCAM. She is a Funded Investigator of SFI research centres, including BiOrbic, CONNECT, I-Form, and funded supervisor in SFI research and training centres ML-labs and D-real. She is a Young Academy Ireland Member.

Nima Afraz (Senior Member, IEEE) received the Ph.D. degree in computer science from Trinity College Dublin, Ireland, in 2020. He is a Tenured Assistant Professor with the School of Computer Science, University College Dublin. He is a Funded Investigator with the CONNECT Research Centre, where his research is focused on open radio access networks, blockchain applications in telecommunications, network economics, and network virtualization. He is a recipient of the Government of Ireland Postdoctoral Fellowship and has worked as a postdoctoral fellow addressing challenges related to the adoption of blockchain technology in telecommunications. He is a Coordinator of an EU MSCA Project RE-ROUTE and has made significant contributions to the Broadband Forum standard TR-402, and the ETSI GS PDL 022 specification. Additionally, he serves as the Vice-Chair of the Linux Foundation's Hyperledger Telecom Special Interest Group.