Sorbonne Université
Ecole Doctorale Informatique, Télécommunications et Electronique de Paris (ED130)
EURECOM
Communication Systems

Programmable Radio Access Network Architecture for Next-Generation Mobile Networks

A Dissertation

Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in the Doctoral School of Sorbonne University

Presented by Chieh-Chun Chen

Thesis defense scheduled for June 20th, 2025, before a jury composed of:

Sorbonne Université
Ecole Doctorale Informatique, Télécommunications et Electronique de Paris (ED130)
EURECOM
Systèmes de Communication

Architecture Programmable du Réseau d'Accès Radio pour les Réseaux Mobiles de Nouvelle Génération

Une Thèse

Présentée en vue de l'obtention partielle du grade de Docteur en Philosophie
au sein de l'École Doctorale de l'Université de la Sorbonne

Présentée par Chieh-Chun Chen

Soutenance prévue le 20 juin 2025, devant un jury composé de:

Rapporteurs	
Prof. TZANAKAKI Anna	Université Nationale et Kapodestrienne d'Athènes, Grèce
Prof. FIORE Marco	Institut des réseaux IMDEA, Espagne
Examinateurs	
Dr. CHOI Nakjung	NOKIA Bell Labs, États-Unis
Dr. Daniel Camps Mur	Fondation i2CAT, Espagne
Prof. KSENTINI Adlen	EURECOM, France
Directeur de Thèse	
Prof. NIKAEIN Navid	EURECOM, France
Invité	
Dr. CHANG Chia-Yu	NOKIA Bell Labs, Belgique

Abstract

5G and future Radio Access Networks (RAN) are evolving toward greater openness and programmability, driven by the principles of Open RAN and initiatives such as Telecom Infra Project (TIP) and O-RAN Alliance. This transformation is enabled by two key factors: (1) open interfaces that ensure vendor-agnostic interoperability, and (2) Software-Defined RAN (SD-RAN) principles, which introduce dynamic and flexible network control. These open interfaces are standardized not only by 3rd Generation Partnership Project (3GPP) but also by the O-RAN Alliance, which plays a crucial role in integrating intelligence into the SD-RAN ecosystem.

In the O-RAN architecture, the SD-RAN controller is defined as the RAN Intelligent Controller (RIC), which enables third parties to develop and deploy SD-RAN applications, known as xApps (for near-real-time control) and rApps (for non-real-time control). Beyond enhancing RAN programmability across different time domains, O-RAN also allows these applications to leverage techniques such as machine learning, enabling more efficient, intelligent, and adaptive network optimization. While O-RAN unlocks broader capabilities, it also presents new challenges. The growing demands of emerging use cases (e.g., real-time gaming, haptic communication) and advanced features (e.g., network slicing) increase complexity, making x/rApps harder to develop and manage. Additionally, due to less standardized interfaces, x/rApps often become tied to specific RIC platforms or vendors, limiting their portability and reuse. Moreover, integrating O-RAN technologies into existing 5G networks further complicates the optimization process, adding burdens for operators.

In this work, we investigate how to enhance the flexibility and programmability of network customization for both the control and user planes of SD-RAN while addressing the complexity introduced by O-RAN technologies, with the goal of simplifying network infrastructure and optimization processes to pave the way for next-generation mobile networks. First, we design a Flexible Control plane (FlexCtrl) that allows control logic to be distributed across three levels — SD-RAN applications, controller, and RAN nodes — based on use case requirements and control loop latency constraints. This design includes an open interface and a virtualization layer for xApps to enable real-time control and simplify development. Second, we propose Integrated and programmable User Plane (IUP), a novel RAN system design with a concrete realization, integrating User Plane Function (UPF) functionalities into a RAN node to enable real-time coordination of traffic management and radio resource allocation. Third, leveraging the synergy of the proposed FlexCtrl and IUP, we introduce AUTO-RAN, an innovative concept that enables autonomous programmability through a robust SD-RAN application design, significantly reducing operational complexity for network operators in RAN optimization while ensuring seamless coordination across both 3GPP and O-RAN ecosystems.

We evaluate the proposed solutions using open-source platforms (OpenAirInterface and FlexRIC). Results demonstrate enhanced flexibility and extended programmability across both the control and user planes. In the control plane, FlexCtrl evolves control logic within SD-RAN applications, simplifying development through recursive abstraction and supporting flexible logic placement to meet use case needs. For example, a distributed control plane achieves control loop latency below 50 µs, suitable for real-time radio scheduling. In the user plane, IUP consolidates traffic management within the RAN, enabling unified control over IP flows and radio resources. This reduces control latency and cuts data delivery overhead by up to 50%. Building on these capabilities, AUTO-RAN integrates autonomous mechanisms into SD-RAN applications, significantly reducing development complexity of x/rApps by up to 90% in line of codes. Results also show that AUTO-RAN achieves real-time adaptability without requiring explicit operator intervention in RAN slicing use cases, and supports intent-driven optimization by allowing operators to declaratively express high-level intents in mobility management use cases.

This thesis simplifies network operations by reducing the development complexity of control logic within RAN functions and SD-RAN applications, streamlining the deployment of network functions in the user plane, and abstracting the intricacies of O-RAN technologies, thereby enabling real-time, unified, and autonomous optimization. Overall, the proposed methods and architectures provide the foundational infrastructure to enhance flexibility, programmability, and customizability across both planes, accelerating the transition toward intelligent next-generation mobile networks.

Résumé

La 5G et les futurs réseaux d'accès radio (RAN) évoluent vers une ouverture et une programmabilité accrues, sous l'impulsion des principes de l'Open RAN et d'initiatives telles que le Telecom Infra Project (TIP) et l'O-RAN Alliance. Cette transformation est rendue possible par deux facteurs clés : (1) des interfaces ouvertes garantissant une interopérabilité indépendante des fournisseurs, et (2) les principes du Software-Defined RAN (SD-RAN), qui introduisent un contrôle réseau dynamique et flexible. Ces interfaces sont normalisées non seulement par le 3rd Generation Partnership Project (3GPP), mais aussi par l'O-RAN Alliance, qui joue un rôle essentiel dans l'intégration de l'intelligence au sein de l'écosystème SD-RAN.

Dans l'architecture O-RAN, le contrôleur SD-RAN est défini comme le contrôleur intelligent RAN (RIC), qui permet à des tiers de développer et de déployer des applications SD-RAN, connues sous les noms de xApps (pour le contrôle en quasi temps réel) et rApps (pour le contrôle en temps non réel). Au-delà de l'amélioration de la programmabilité du RAN dans différents domaines temporels, O-RAN permet également à ces applications d'exploiter des techniques telles que l'apprentissage automatique, rendant l'optimisation du réseau plus efficace, intelligente et adaptative. Si O-RAN ouvre de nouvelles perspectives, il soulève également plusieurs défis. Les exigences croissantes des cas d'usage émergents (par exemple, les jeux en temps réel ou la communication haptique), ainsi que des fonctions avancées comme le découpage du réseau (network slicing), augmentent la complexité et rendent le développement et la gestion des x/rApps plus difficiles. Par ailleurs, en raison d'interfaces encore peu standardisées, les x/rApps deviennent souvent dépendantes de plateformes RIC ou de fournisseurs spécifiques, limitant ainsi leur portabilité et leur réutilisabilité. Enfin, l'intégration des technologies O-RAN dans les réseaux 5G existants complique davantage le processus d'optimisation, ce qui accroît la charge opérationnelle pour les opérateurs.

Dans ce travail, nous étudions comment améliorer la flexibilité et la programmabilité de la personnalisation du réseau pour les plans de contrôle et utilisateur du SD-RAN, tout en abordant la complexité introduite par les technologies O-RAN, dans le but de simplifier l'infrastructure réseau et les processus d'optimisation, afin de préparer la transition vers les réseaux mobiles de nouvelle génération. Premièrement, nous concevons un plan de contrôle flexible (FlexCtrl) qui permet de distribuer la logique de contrôle sur trois niveaux — les applications SD-RAN, le contrôleur, et les nœuds RAN — en fonction des exigences des cas d'usage et des contraintes de latence des boucles de contrôle. Cette conception comprend une interface ouverte et une couche de virtualisation pour les xApps, permettant un contrôle en temps réel tout en simplifiant le développement. Deuxièmement, nous proposons un plan utilisateur intégré et programmable (IUP), une nouvelle conception du système RAN avec une réalisation concrète, intégrant les fonctionnalités de la User Plane Function (UPF) dans un nœud RAN afin de permettre la coordination en temps réel de la gestion du trafic et de l'allocation des ressources radio. Troisièmement, en tirant parti de la synergie entre FlexCtrl et IUP, nous présentons AUTO-RAN, un concept innovant qui permet une programmabilité autonome grâce à une conception robuste d'application SD-RAN. Cette approche réduit significativement la complexité opérationnelle pour les opérateurs dans l'optimisation du RAN, tout en assurant une coordination transparente entre les écosystèmes 3GPP et O-RAN.

Nous évaluons les solutions proposées en nous appuyant sur des plateformes open source (OpenAirInterface et FlexRIC). Les résultats démontrent une flexibilité accrue et une programmabilité étendue des plans de contrôle et d'utilisateur du SD-RAN. Dans le plan de contrôle, FlexCtrl fait évoluer la logique de contrôle au sein des applications SD-RAN, en simplifiant le développement grâce à une abstraction récursive, tout en permettant un placement flexible de la logique selon les besoins des cas d'usage. Par exemple, un plan de contrôle distribué permet d'atteindre une latence de boucle de contrôle inférieure à 50 µs, adaptée à la planification radio en temps réel. Dans le plan utilisateur, IUP consolide la gestion du trafic au sein du RAN, permettant un contrôle unifié des flux IP et des ressources radio. Cette approche permet de réduire la latence de contrôle et de diminuer les frais de transmission des données jusqu'à 50%. En s'appuyant sur ces capacités, AUTO-RAN intègre des mécanismes autonomes dans les applications SD-RAN, réduisant significativement la complexité du développement des x/rApps, jusqu'à 90% en nombre de lignes de code. Les résultats montrent également qu'AUTO-RAN permet une adaptabilité en temps réel sans intervention explicite de l'opérateur dans les cas d'utilisation de découpage du RAN, et qu'il prend en charge l'optimisation basée sur les intentions en permettant aux opérateurs d'exprimer de manière déclarative des intentions de haut niveau dans les cas d'utilisation de gestion de la mobilité.

Cette thèse simplifie les opérations réseau en réduisant la complexité du développement de la logique de contrôle au sein des fonctions RAN et des applications SD-RAN, en rationalisant le déploiement des fonctions réseau dans le plan utilisateur, et en abstrahant les complexités des technologies O-RAN, permettant ainsi une optimisation en temps réel, unifiée et autonome. Dans l'ensemble, les méthodes et architectures proposées fournissent une infrastructure de base visant à renforcer la flexibilité, la programmabilité et la personnalisation sur les deux plans, accélérant ainsi la transition vers les réseaux mobiles intelligents de nouvelle génération.

Acknowledgements

This thesis marks the end of a long and challenging journey, one I could not have completed without the support, guidance, and encouragement of many remarkable people. First, I am deeply thankful to my supervisor, Navid Nikaein, for believing in me and offering the opportunity to pursue a PhD in such a technically demanding and forward-looking field. His deep expertise in mobile networks, combined with his openness to new ideas and his constant encouragement, played a key role in shaping both the direction and the impact of my research. I also appreciate the many opportunities he provided to present our work at workshops, conferences, and industry events, which helped me build confidence, gain visibility, and develop a stronger connection to the broader research and industrial community.

I would also like to express my heartfelt thanks to Chia-Yu Chang, who supported me from the very beginning, especially when I was learning how to write my first research papers. His thoughtful feedback, patience, and collaboration over the years have been a true foundation for my academic growth. My sincere thanks go to Alireza Mohammadi, Ilias Chatzistefanidis, and Mikel Irazabal for the insightful discussions and collaborations that helped me navigate many difficult phases of the research. Their perspectives often clarified the challenges I faced and opened up new directions. I am equally thankful to Nguyen Duc Khai, Yueh-Huan Chung, and Setareh Alagheh Band for their valuable contributions during their internships. It was a pleasure to work with such capable and dedicated individuals.

Beyond the research itself, my experience at EURECOM was made truly enjoyable thanks to the vibrant and supportive community. I feel fortunate to have shared this journey with the BubbleRAN team, the OpenAirInterface community, my office mates, and the many colleagues and friends I met during my PhD. Finally, I want to express my deepest gratitude to my family and to Andrew Kearns. Your constant support, encouragement, and presence gave me strength through every high and low of this journey. I am sincerely thankful to you all.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

1.1 Evolution of Mobile Network Architecture

Over the past two decades, mobile networks have undergone a significant evolution, transitioning from Third Generation (3G) to Fourth Generation (4G), and now Fifth Generation (5G), fundamentally reshaping global connectivity. Each generation has introduced key technological advancements that have expanded network capabilities, improved performance, and unlocked new applications across industries, as shown in Table 1.1.

The introduction of 3G in the early 2000s marked a major shift in mobile communication. It was the first generation to support mobile internet access, transitioning from circuit-switched to packet-switched architecture for data services. This shift enabled more devices to connect to the internet, facilitating mobile web browsing, email, and basic multimedia streaming. Standardized by the 3rd Generation Partnership Project (3GPP), 3G technologies - such as Wideband Code Division Multiple Access (WCDMA) and High-Speed Packet Access (HSPA) - laid the groundwork for modern mobile networks, providing the essential infrastructure for future advancements [138].

Building upon the foundation established by 3G, the 2010s saw the introduction of 4G, with Long-Term Evolution (LTE) and LTE-Advanced (LTE-A), bringing significant enhancements in speed, capacity, and network efficiency. Unlike 3G, which was the first to introduce mobile internet access, 4G fully transitioned to an all-IP-based architecture [126, 138], removing circuit-switched communication in the Core Network (CN). This shift enabled much higher data speeds (100 Mbps to 1 Gbps) and improved support for a rapidly growing number of connected devices. Moreover, traditional voice and text services were replaced by VoLTE, ensuring seamless integration with modern digital services (e.g., iMessage, WhatsApp). With these advancements, 4G became the backbone of the modern digital world, enabling the rise of social media, streaming, online shopping, and digital finance. Its improved efficiency enabled increasingly demanding applications, paving the way for 5G. While the CN architecture underwent a major transformation, the Radio Access Network (RAN) architecture saw more limited advancements from 3G to 4G. Although 4G introduced standardized interfaces (e.g., S1 interface, which allowed CN and RAN to be sourced from different vendors), most deployments remained vendor-specific. Advanced RAN features — such as Software-Defined RAN (SD-RAN), Virtualized RAN (vRAN), Centralized RAN (C-RAN), and RAN Controller — were implemented primarily through proprietary vendor solutions, limiting multi-vendor interoperability and scalability.

Table 1.1: Evolutions of Mobile network architecture.

Generation	**RAN Architecture**			**CN Architecture**	**Services**		
	Monolithic	**Disaggregated**	**RAN Control & Optimization**		**Voice Call**	**SMS Text**	**Internet Data**
2G	Vendor Locked	-	Vendor-Locked Base Station Controller	Circuit-Switched	✓	✓	-
3G	Vendor Locked	-	Vendor-Locked Radio Network Controller	Circuit-Switched & Packet-Switched	✓	✓	✓ (Low Data Rate)
4G	Vendor Specific	Vendor Specific (vRAN, C-RAN)	Vendor Specific RAN Controller	All-IP-Based	✓(VoLTE)		✓ (High Data Rate)
5G with Open RAN	Vendor Specific	Vendor Agnostic (CU, DU, RU)	Vendor Agnostic RAN Intelligent Controller	Service-Based	✓(VoNR)		✓ (eMBB, uRLLC, mMTC)

As mobile demands grew, the 2020s marked the arrival of 5G New Radio (NR), driven by the need for greater network efficiency, enhanced service assurance, and architectural flexibility. Unlike its predecessors, 5G introduced several key innovations toward an application-aware network design, incorporating standardized capabilities such as tailored Quality of Service (QoS) [18], network slicing [5], and a more flexible network architecture [8]. These advancements allow network operators to dynamically customize and configure network deployments, optimizing performance for a variety of use cases. To leverage these innovations, 5G is structured around three primary service categories, each targeting distinct performance needs: (1) enhanced Mobile Broadband (eMBB) – enabling cloud gaming, high-definition streaming, and immersive Augmented/Virtual Reality (AR/VR) experiences, (2) ultra-Reliable Low-Latency Communications (uRLLC) – supporting autonomous vehicles, remote surgery, and mission-critical applications, (3) massive Machine-Type Communications (mMTC) – enabling smart cities and industrial automation.

Delivering these services efficiently requires a highly adaptable network architecture. To achieve this, 5G introduces an evolved architecture that enhances both CN and RAN. In the CN, 5G adopts a Service-Based Architecture (SBA) by standardizing more interfaces between network functions, enabling greater flexibility in deploying network functions based on service requirements. For example, User Plane Functions (UPFs) can be distributed across access, edge, and cloud sites, optimizing latency, network efficiency, and application performance based on specific use cases. In contrast, 5G RAN has fewer standardized interfaces, though some, like the F1 interface between Distributed Units (DUs) and Centralized Units (CUs), allow for more scalable and flexible deployments. This enables a single CU to manage multiple DUs, improving scalability and resource utilization. However, many of these implementations remain vendor-specific, much like in 4G, making RAN deployments largely proprietary and restricting openness in mobile network infrastructure.

As 5G adoption accelerates, the lack of interoperability in RAN has become a major challenge, limiting deployment flexibility and increasing costs for operators. To address this challenge, the concept of Open RAN [84, 127, 147] has emerged as a transformative paradigm, aiming to break traditional vendor lock-in and promote interoperability between network components from different vendors. A key driver of this movement is the O-RAN Alliance [36], which actively defines open interfaces and specifications based on 3GPP standards. By doing so, it fosters a more flexible, cost-efficient, and scalable network environment, enabling operators to mix and match RAN components, such as CU, DU and Radio Unit (RU), from different vendors while optimizing performance and cost. Within the O-RAN architecture [117], open interfaces are introduced across various network elements, including O1 interface for Service Management and Orchestration (SMO) [38, 39], as well as A1 and E2 interfaces for RAN Intelligent Controllers (RICs) [29, 30, 40, 42]. Building on SD-RAN principles, which separate the Control and User Planes (CP and UP) functions, O-RAN further enhances programmability by enabling RICs to provide centralized, intelligent control of RAN nodes across multiple vendors through standardized open interfaces. As a key component of O-RAN, the RICs, comprising the Near-Real and Non-Real Time RICs (NearRT-RIC [40] and NonRT-RIC) [30]), act as the SD-RAN controllers, optimizing RAN operations through their control applications (e.g., xApps in NearRT-RIC and rApps in NonRT-RIC), while also facilitating the integration of advanced technologies such as Artificial Intelligence (AI) and Machine Learning (ML).

With the industry moves beyond 5G and toward 6G, RAN architecture must evolve to become more flexible, programmable, and intelligent to support application-centric networks. O-RAN (or Open RAN) is expected to play a central role in shaping the next-generation mobile networks by fostering an open, interoperable, and software-driven ecosystem, reducing reliance on proprietary solutions. Integrating O-RAN architectures into existing infrastructures unlocks new opportunities for real-time programmability, AI/ML-driven optimizations, and seamless orchestration of diverse network functions (e.g., intelligent control applications). These advancements will enhance network scalability and efficiency while enabling emerging use cases like ubiquitous connectivity, autonomous systems, and immersive digital experiences, driving the evolution of future mobile networks.

1.2 Challenges

Realizing and improving flexibility to enhance greater programmability and customizability of RAN introduces several challenges, spanning technical, operational, and economic dimensions. Technically, achieving a flexible control mechanism that supports not only near- and non-real-time control operations but also real-time control for time-sensitive RAN processes remains a significant hurdle in O-RAN architecture. Additionally, coordinating 3GPP-defined UP functions across both CN and RAN domains is a key challenge in optimizing both network and user performance in future application-centric mobile networks. From an operational and economic perspective, integrating O-RAN technologies with existing 3GPP infrastructure while managing its complexities is crucial to ensuring seamless interoperability and widespread adoption. This section discusses these challenges and explores potential solutions for building a more flexible and programmable RAN architecture for next-generation mobile networks. Figure 1.1 illustrates these challenges within their respective domains alongside the corresponding contributions presented in this thesis.

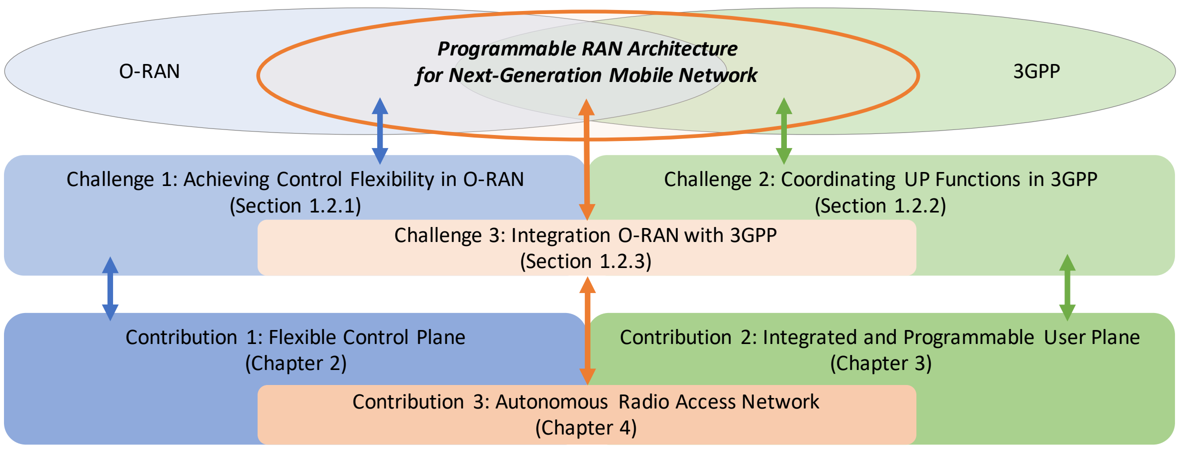

O-RAN
Programmable RAN Architecture for Next-Generation Mobile Network
3GPP
Challenge 1: Achieving Control Flexibility in O-RAN (Section 1.2.1)
Challenge 2: Coordinating UP Functions in 3GPP (Section 1.2.2)
Challenge 3: Integration O-RAN with 3GPP (Section 1.2.3)
Contribution 1: Flexible Control Plane (Chapter 2)
Contribution 2: Integrated and Programmable User Plane (Chapter 3)
Contribution 3: Autonomous Radio Access Network (Chapter 4)

Figure 1.1: Addressed challenges and corresponding contributions in enabling a programmable RAN architecture for next-generation mobile networks.

1.2.1 Achieving Control Flexibility in O-RAN

O-RAN introduces vendor-agnostic interoperability through standardized open interfaces as well as standardized Service Models (SMs), such as Key Performance Measurement (KPM) [41], RAN Control (RC) [46], and Cell Configuration Control (CCC) [44]. Additionally, it enables control operations in both near- and non-real-time domains, allowing adaptation to different network scenarios. While these advancements enhance interoperability and flexibility, they also introduce significant development complexities for both control applications (e.g., xApps) and RAN functions, making real-time control more challenging to implement. A major limitation of current O-RAN architectures is that control logic is statically confined to the control application layer, further increasing the complexity of control application development. This lack of flexibility restricts the ability to dynamically adjust control logic based on network conditions and specific scenario requirements (e.g., dynamic radio resource scheduling).

1.2.1.1 Complexity of Control Application

One of the key challenges in achieving control flexibility is the increasing complexity of xApp development in the O-RAN architecture, where xApps operate within the NearRT-RIC. This complexity limits xApp real-time control capabilities and hinders its reusability and portability in multi-vendor environments. It primarily stems from the rigid structure of O-RAN-defined SMs and the need for xApps to interact with heterogeneous RAN nodes, each with vendor-specific implementations, further increasing development difficulties.

As xApp control logic becomes more sophisticated, the challenge intensifies due to the need to process large volumes of data from multiple SMs across different RAN nodes. Parsing and transforming complex data structures introduces significant processing overhead, making real-time optimization increasingly difficult (cf. Section 2.3.3). Additionally, as xApp complexity increases, maintaining low-latency decision-making becomes more challenging, further limiting the feasibility of real-time control operations.

Beyond software challenges, the lack of a standardized interface between xApps and the NearRT-RIC further hinders portability (cf. Section 2.3.1). Without a unified framework, xApps must be customized for vendor-specific NearRT-RIC Application Protocol Interfaces (APIs) [118], leading to compatibility issues and increased development overhead. This fragmentation restricts deployment flexibility and limits scalability in multi-vendor environments.

Moreover, the rigid structure of standardized SMs constrains functional extensibility, making it difficult to support emerging network requirements. For example, the RC SM lacks support for application flow control in scenarios requiring Low Latency, Low Loss, Scalable Throughput (L4S) [62], preventing xApps from optimizing such applications. This limitation forces developers to implement workarounds or vendor-specific extensions, compounding fragmentation within the Open RAN ecosystem.

Summary and Potential Solutions. These challenges increase development time and costs while slowing innovation, as xApp developers need to focus on adapting to vendor-specific constraints rather than advancing higher-level network optimization strategies. Without addressing these barriers, xApp innovation will remain limited, restricting the implementation of a flexible control mechanism capable of supporting real-time control operations. Overcoming these challenges will require solutions that enable vendor-agnostic development and streamline xApp design to improve control flexibility.

1.2.1.2 Complexity of RAN Function

Another key challenges in achieving control flexibility is the increasing complexity of RAN function development, which requires enabling programmability to ensure that RAN functions can efficiently execute control actions based on decisions made by higher-layer x/rApps. For example, to meet Service Level Agreements (SLAs), a network operator may deploy an xApp to create multiple network slices within the RAN, requiring the RAN to dynamically allocate resources for each slice. However, enabling this capability demands that RAN vendors develop slicing-related functions, incorporating specialized algorithms and technologies for slice creation and resource allocation.

Additionally, RAN function development has to integrate logic from O-RAN-defined SMs to correctly interpret control messages sent from xApps. This means implementations need to be capable of decoding O-RAN-defined messages and interfacing with internal RAN APIs to execute control actions. As the number of O-RAN SMs continues to grow and their structures become increasingly complex, the development and integration challenges in RAN systems are further amplified.

Beyond handling control decisions from x/rApps, real-time decision-making introduces additional complexity. Many time-sensitive RAN processes, such as radio resource scheduling and beam management, require ultra-low latency responses and efficient processing, which centralized control applications often struggle to provide. To address this limitation, some approaches (e.g., dApp [81] and JANUS [83]) enable real-time programmability by executing control operations locally. While embedding control logic within the RAN reduces control latency, it also significantly increases the complexity of RAN function development (cf. Section 2.3.2), necessitating additional mechanisms for dynamic control adjustments, conflict resolution, and multi-cell coordination.

However, the complexity of RAN functions is further exacerbated by vendor inconsistencies in O-RAN deployments. RAN vendors implement O-RAN SMs and internal control logic differently, leading to interoperability issues. For example, an xApp sending the same control message to multiple RAN nodes may encounter varying levels of support for the requested control actions. Consequently, xApp development becomes increasingly complex, as it has to accommodate these variations and implement additional logic to ensure compatibility across diverse RAN implementations (cf. Section 2.3.3).

Summary and Potential Solutions. In summary, RAN function development complexity arises from the need to enable programmable functionalities, handle control decisions from x/rApps, integrate O-RAN SM logic, and support real-time decision-making within the RAN. These challenges not only impact real-time network operations but also introduce vendor-specific inconsistencies, making standardization and large-scale O-RAN adoption more difficult. To overcome these barriers, solutions should enhance control mechanisms and optimize processing frameworks to enable flexible, adaptive control across near-, non-, and real-time latency requirements, ensuring a scalable and programmable RAN architecture for future mobile networks.

1.2.2 Coordinating UP Functions in 3GPP

As mobile networks evolve, newer generations of 3GPP, along with standards issued by ITU and IEEE, are shifting from a network-centric to an application-centric approach [104, 108, 146]. In this paradigm, the network needs to provide an extensive and ubiquitous medium for serving diverse application requirements. However, the current RAN architecture faces significant challenges in adapting to this shift. Two primary issues emerge: (1) disparities in UP function management across CN and RAN and (2) the complexity of distributed UP functions.

1.2.2.1 Disparities Between CN and RAN

One of the key challenges in coordinating UP functions is the increasing disparity in UP function management between CN and RAN. As mobile networks evolve toward application-centric architectures, the lack of coordination between UP functions across CN and RAN creates significant challenges in meeting the diverse requirements of emerging applications such as Augmented Reality (AR) and Virtual Reality (VR).

Despite ongoing research efforts, most existing solutions focus on optimizing either CN or RAN performance in isolation rather than addressing coordination between the two. For instance, several works [53, 72, 98, 133] primarily enhance RAN performance, improving radio resource scheduling efficiency. Similarly, other approaches [60, 89, 110] focus on CN performance, improving packet processing efficiency. However, without effective coordination, these independent optimizations create disparities that make end-to-end performance optimization increasingly difficult.

A major source of disparity occurs when data transitions from the IP-based Data Network (DN) to the mobile network. This process involves multiple transformations that can introduce inefficiencies. In the CN, the User Plane Function (UPF) aggregates and classifies IP flows into one or more QoS flows based on predefined parameters such as 5QI values [18]. These QoS flows are then mapped to Data Radio Bearers (DRBs) in the RAN by the Service Data Adaptation Protocol (SDAP) layer, which further reduces the granularity of the original IP flows. This transformation introduces inconsistencies in how traffic is handled across different network domains (cf. Section 3.3.1).

The lack of seamless UP coordination affects network performance in multiple ways. First, as data moves from the DN to CN, the process of mapping IP flows into QoS flows may misrepresent application traffic characteristics, leading to inefficient flow prioritization. This issue is particularly problematic in applications requiring fine-grained traffic differentiation, such as ultra-low latency communications and real-time media streaming. Second, as traffic moves from the CN to RAN, these disparities impact the ability of RAN to allocate adequate resources, preventing it from meeting the stringent requirements of next-generation mobile applications.

Although 3GPP has defined a set of QoS characteristics[18] and network slicing mechanisms[5] to ensure differentiated traffic treatment, the effectiveness of these solutions depends on vendor-specific implementations. Different vendors may implement QoS mapping, flow classification, and resource allocation mechanisms differently, further intensifying inconsistencies between CN and RAN, especially in Open RAN environments. As a result, programmability and adaptability to diverse application behaviors remain limited, making it increasingly difficult to support emerging mobile network use cases (cf. Section 3.3.2).

Summary and Potential Solutions. Optimizing end-to-end performance requires addressing disparities in flow management and resource allocation between the CN and RAN. Seamless UP coordination is crucial for supporting the application-centric evolution of mobile networks and ensuring consistent performance management across network domains. A unified, programmable RAN architecture can bridge these gaps by enabling software-defined control over UP functions, allowing dynamic network policy adaptation and resource allocation to meet diverse application requirements.

1.2.2.2 Complexity in Distributed UP Functions

Another key challenge in coordinating UP functions is the growing complexity of distributed UP functions in mobile network infrastructure. In 5G and beyond, network deployment has become more flexible, allowing network operators to customize deployments based on specific service requirements. For example, UPFs can now be deployed across cloud, edge, and access sites, depending on service latency constraints. Similarly, with the emergence of Open RAN, the disaggregated RAN architecture enables operators to mix and match network components (e.g., RU, DU, and CU) based on resource availability and coverage needs.

While these advancements enhance deployment flexibility, they also increase operational complexity. The rise in distributed network components requires operators to manage, configure, and optimize a growing number of elements, making network orchestration more challenging. Additionally, interworking between 3GPP and non-3GPP networks requires additional gateway hops [1], further complicating infrastructure and increasing processing overhead (cf. Section 3.3.3).

The modern RAN architecture must efficiently handle IP-based traffic, unlike its original design for circuit-switched voice and text services. However, within the end-to-end UP, RAN operates as a heavy Layer 2 entity in the OSI model, which restricts IP packet traversal efficiency. To address this obstacle, various optimizations have been introduced to improve UPF performance[60, 89, 110] and reduce transport latency by shortening the physical distance between UPF and RAN or DN and UPF [125].

However, these optimizations primarily focus on improving RAN backhaul performance, often overlooking coordination with the radio link. As a result, latency issues persist, particularly due to the placement of UPF components across different network segments and the overhead introduced by GPRS Tunneling Protocol User Plane (GTP-U) tunneling. The continued reliance on GTP-U tunnels complicates traffic flow optimization between RAN and CN, adding protocol processing overhead and further increasing latency (cf. Section 3.3.1). This issue is especially critical in emerging technologies such as Open RAN, where additional UP hops further exacerbate latency challenges.

Summary and Potential Solutions. As mobile networks evolve, an efficient, programmable, and well-coordinated RAN architecture becomes increasingly essential. Addressing the complexity of distributed UP functions and enhancing UP coordination between the CN and RAN are key to optimizing network and user performance, ensuring low-latency, high-throughput connectivity for future mobile applications.

1.2.3 Abstracting O-RAN Complexities in 3GPP Networks

As O-RAN emerges within the 3GPP-based mobile network infrastructure, it offers new opportunities for vendors and operators by enabling greater customizability. Through open interfaces, operators can deploy networks with more flexibility while leveraging RAN programmability via the RIC to support new use cases. However, despite these advantages, integrating O-RAN into 3GPP-based systems introduces significant challenges due to the complexity of managing dual ecosystems.

While O-RAN specifications are built upon 3GPP standards, they focus primarily on RAN optimizations and lack direct coordination with CN functions. This architectural gap affects network operators in multiple ways. First, it complicates end-to-end network optimization, as O-RAN lacks built-in mechanisms to coordinate with CN functions such as UPF. This fragmentation forces operators to develop custom solutions, increasing integration costs and operational overhead.

Second, the adoption of O-RAN-specific frameworks (e.g., Service Management Orchestration [117]), tools (e.g., gRPC [122]), and architectures (e.g., O-Cloud [117]) further increases network complexity. Unlike traditional 3GPP-based systems, where network functions are tightly integrated, O-RAN's modular approach requires additional configuration and adaptation efforts, placing a heavier burden on network operators.

Finally, the lack of seamless integration of control applications limits real-time adaptability, making it challenging to implement dynamic network adjustments. Even though vendors provide rApps with ML-driven network slicing decisions and xApps with real-time control for slice modification, operators often lack the necessary O-RAN expertise to deploy and integrate them effectively. This slows the adoption of advanced network optimization techniques, such as control applications for load balancing management, interference management, and energy-saving mechanisms.

Summary and Potential Solutions. Consequently, these complexities not only increase deployment overhead but also hinder real-time adaptability, slowing the seamless adoption of advanced RAN programmability. Addressing this challenge is essential for enabling a programmable RAN architecture while ensuring the effective integration of O-RAN and leveraging its capabilities for next-generation mobile networks.

1.3 Motivation and Problem Statement

Building on the challenges outlined above, achieving a programmable RAN architecture is essential for enabling the next generation of adaptable and intelligent mobile networks. This section presents the key motivations driving the need for a programmable RAN and defines the specific research problems addressed in this thesis.

1.3.1 Why Programmable RAN Architecture?

A programmable RAN unlocks new capabilities that enhance network flexibility for real-time adaptability, enable efficient coordination for end-to-end network optimization, and ensure smooth interoperability between heterogeneous ecosystems (e.g., O-RAN and 3GPP). These capabilities are critical for supporting evolving application-driven services and optimizing network efficiency at scale.

Enhancing Network Flexibility and Adaptability. Modern mobile networks must support a wide range of deployment scenarios, such as network slicing and multi-cell coordination. However, traditional RAN architectures, including early SD-RAN implementations, often lack the agility needed for real-time adaptability due to centralized control structures and vendor-specific constraints. A programmable RAN introduces software-defined control mechanisms, enabling networks to adjust in real time to meet changing deployment requirements. This flexibility allows operators to customize control mechanisms, dynamically allocate radio resources, optimize network policies, and fine-tune performance across different time scales — without disrupting RAN operations. As mobile networks become increasingly complex, real-time optimizations will be essential for ensuring performance, reliability, and seamless service delivery in next-generation mobile systems.

Enabling Consistent End-to-End Network Optimization. Modern applications such as immersive XR, autonomous vehicles, and industrial automation demand ultra-low latency, high throughput, and intelligent resource management to deliver seamless user experiences. A programmable RAN plays a crucial role in enhancing coordination between RAN and CN functions, particularly in UP operations. By dynamically managing UP resources across the entire network stack, operators can make real-time adjustments to traffic routing, load balancing, and QoS management. A software-defined approach enables demand-driven resource allocation, ensuring consistent management across network domains. This level of end-to-end coordination improves traffic management, enhances application performance, and optimizes overall network efficiency, making programmable RAN a cornerstone of future application-centric mobile networks.

Enabling Seamless Interoperability Between O-RAN and 3GPP Networks. A programmable RAN drives innovation by embracing openness, interoperability, and vendor-agnostic deployment models inspired by Open RAN principles. As O-RAN adoption expands, mobile networks gain the ability to accelerate innovation cycles and improve cost efficiency while maintaining compliance with 3GPP standards. By seamlessly integrating O-RAN technologies with existing 3GPP infrastructure, a programmable RAN enables autonomous control mechanisms for intelligent network optimization. This programmability empowers operators to deploy new technologies more efficiently, implement AI-driven optimizations, and enhance interoperability across diverse network components— without increasing operational overhead. This shift toward autonomous, intelligent, and highly interoperable networks provides operators with greater control, flexibility, and scalability, paving the way for a more agile, future-ready mobile ecosystem.

1.3.2 Research Problem Definition

Despite its potential, achieving a fully programmable RAN architecture presents fundamental research problems that need to be addressed to realize its full potential for next-generation mobile networks. This thesis focuses on three key problems:

1.3.2.1 The Problem of Enabling a Flexible Control Mechanism

- **Challenge:** Developing control applications and RAN functions remains complex, restricting the ability to implement a flexible and programmable control mechanism.
- **Motivation:** Simplifying control application design is essential for enabling real-time adaptability in RAN operations, while simplifying RAN function development is crucial for achieving a more adaptable, scalable, and programmable control mechanism.
- **Research Question:** How can a flexible and programmable control mechanism be achieved while reducing the complexity of control applications and RAN functions, and ensuring real-time control capabilities?

1.3.2.2 The Problem of Enabling Coordination UP functions between CN and RAN

- **Challenge:** The distributed nature of UP functions complicates end-to-end network optimization and increases infrastructure complexity, making it difficult to achieve seamless coordination across CN and RAN.

- **Motivation:** Achieving seamless coordination of UP functions across CN and RAN is essential for efficient flow control, optimized radio resource allocation, and consistent network management.
- **Research Question:** How can UPF in the CN and UP functions in the RAN be effectively controlled and coordinated in real-time without introducing additional infrastructure complexity?

1.3.2.3 The Problem of Integrating O-RAN into 3GPP Networks

- **Challenge:** O-RAN introduces architectural complexities that make it difficult for network operators to manage and integrate with existing 3GPP networks.
- **Motivation:** Abstracting O-RAN complexities will enable operators to seamlessly adopt and leverage O-RAN technologies (xApps, rApps) within 3GPP networks, without increasing operational overhead.
- **Research Question:** How can network operators leverage O-RAN technologies within existing 3GPP networks while minimizing operational complexity?

1.4 Thesis Contributions and Structure

In this thesis, we investigate both O-RAN and 3GPP networks to enhance the customizability and programmability of RAN, with the goal of realizing a programmable RAN architecture that can adapt to future application-centric networks. The thesis begins with an introduction to the evolution of mobile network architectures in Section 1.1, providing a historical and technical foundation. Additionally, background and related work on relevant concepts and technologies related to O-RAN and 3GPP are incorporated into the initial sections of each chapter.

Building upon the research challenges and problems discussed in Sections 1.2 and 1.3, this thesis is structured to systematically address each key issue. Each chapter focuses on a specific research problem, and Figure 1.1 illustrates how the contributions align with the identified research challenges. The key contributions of this thesis are as follows:

Chapter 2: Flexible Control Plane. This chapter introduces FlexCtrl, a flexible CP architecture designed to address the challenges of control flexibility and real-time programmability in O-RAN and SD-RAN. FlexCtrl supports all three control topologies — centralized, decentralized, and distributed — enabling adaptable control logic placement to accommodate different deployment scenarios. To tackle the complexity of xApp development and real-time control capabilities, we propose FlexApp, a framework within the decentralized topology of FlexCtrl. FlexApp provides native support for third-party xApps through its E2* interface, addressing current limitations in portability by enabling interoperability between Near-RT RIC platforms and third-party xApps, while also supporting ultra-low-latency control operations (<10 ms). To further abstract the complexity of O-RAN SMs tied to RAN Functions (RFs), FlexApp enables recursive virtualization of RFs. This high-level abstraction alleviates the rigidity of standardized SMs and makes it easier for xApps to control RFs across vendor-specific implementations. Additionally, we redesign the radio resource scheduler, introducing a recursive scheduler for multi-level resource allocation, to demonstrate FlexCtrl's capability in controlling RAN functions in real-time.

Chapter 3: Integrated and Programmable User Plane. This chapter introduces IUP, a novel RAN system to address the lack of coordination between UP functions across CN and RAN and the growing complexity of distributed UP functions in mobile network infrastructure. IUP unifies UP functions across the RAN and UPF into a single programmable entity, envisioned as a foundational component for next-generation mobile networks. By integrating UP functions, IUP simplifies mobile network architecture, reducing latency and protocol overhead (e.g., GTP-U) while enabling end-to-end programmability — from IP flow traffic control to radio resource allocation. This application-centric design ensures the network can dynamically adapt to diverse service requirements. To tackle the complexity of distributed UP functions, IUP introduces the Integrated Data Flow Control (IDFC) sublayer, a new traffic management pipeline integrated into the RAN protocol stack. Additionally, programmable rules for IP traffic control and radio resource allocation provide greater programmability, enabling simultaneous control of both traffic flow and radio resources. Furthermore, IUP facilitates seamless interworking with non-3GPP networks, enhancing interoperability across heterogeneous network domains. Several key use cases, including handover and roaming, are analyzed to demonstrate IUP's compatibility with existing deployments and its ability to simplify complex mobile network infrastructures. Finally, real-world testbed results validate IUP's practical benefits, including reduced network latency and overhead, seamless convergence between 3GPP and non-3GPP networks, and enhanced programmability for IP traffic control and radio resource allocation to support diverse applications.

Chapter 4: Autonomous Radio Access Network. This chapter introduces AUTO-RAN, a novel concept that abstracts the current O-RAN architecture with an autonomous control loop, addressing the challenges of integrating O-RAN into 3GPP-based mobile networks. By simplifying optimization process, AUTO-RAN enables operators to improve both network and user performance in a declarative manner - without needing to manage the complexity of two distinct architectures. AUTO-RAN allows operators to interact with the network using familiar 3GPP-defined logic, while seamlessly leveraging the programmability of O-RAN. This approach ensures interoperability between the two ecosystems and supports real-time adaptability, enhancing overall performance without requiring operators to directly intervene in O-RAN's internal mechanisms. Furthermore, AUTO-RAN facilitates the smooth integration of new features introduced by either 3GPP or O-RAN, thanks to its foundation across both standards. Finally, we demonstrate its effectiveness through real-world use cases, highlighting how AUTO-RAN automatically optimizes network and service performance via robust SD-RAN applications integrated with autonomous control loop mechanisms. This implementation underscores AUTO-RAN's role in bridging the two architectures, making network operations more autonomous, adaptive, and efficient.

Finally, Table 1.2 presents selected publications and demonstrations in chronological order, highlighting their contributions to the respective thesis chapters.

Table 1.2: Categorization of publications, demonstrations, and their contributions to specific chapters.

Type	Reference	Ch. 2	Ch. 3	Ch. 4
Workshop Demo	Chieh-Chun Chen, Navid Nikaein, and Mikel Irazabal Bengoa. "FlexRIC: an SDK for next-generation SD-RANs-slicing and traffic control use-case". In: *Summer OpenAirInterface Workshop, 12-13 July 2022, Paris, France* (2022). URL: https://youtu.be/sHJSA3FgGd8	✓	✓	✓
Conference Paper	Chieh-Chun Chen et al. "FlexApp: Flexible and Low-Latency xApp Framework for RAN Intelligent Controller". In: *ICC 2023 - IEEE International Conference on Communications.* 2023, pp. 5450–5456. DOI: 10.1109/ICC45041.2023.10278600	✓	✓	✓
Exhibition Demo	Chieh-Chun Chen, Alireza Mohammadi, and Navid Nikaein. "xApp independent lifecycle, Interactive RAN Slicing xApp for xApp maintainer from Open RAN operator". In: *Mobile World Congress, 27 February - 2 March 2023, Barcelona, Spain* (2023). URL: https://youtu.be/kbh31hSxVFI	✓		
Workshop Demo	Chieh-Chun Chen et al. "xApp DevOps evolution and observable OAM in open RAN ecosystem". In: *Joint OSC/OSFG-OAI Workshop: End-to-End Reference Designs for O-RAN, 14-15 November 2023, Burlington, MA, USA* (2023). URL: https://www.eurecom.fr/publication/7863	✓		✓
Conference Paper	Chieh-Chun Chen, Chia-Yu Chang, and Navid Nikaein. "FlexSlice: Flexible and real-time programmable RAN slicing framework". In: *GLOBECOM 2023 - 2023 IEEE Global Communications Conference.* 2023, pp. 3807–3812. DOI: 10.1109/GLOBECOM54140.2023.10437791	✓	✓	
Exhibition Demo	Chieh-Chun Chen, Navid Nikaein, and Alireza Mohammadi. "ECO-RAN: Multi-cell mobility management in an end-to-end 5G O-RAN network". In: *Mobile World Congress, 26-29 February 2024, Barcelona, Spain* (2024). URL: https://youtu.be/hlLt--WSQPc	✓		✓
Magazine Paper	Chieh-Chun Chen, Chia-Yu Chang, and Navid Nikaein. "IUP: Integrated and Programmable User Plane for Next-Generation Mobile Networks". In: *IEEE Network* (2025), pp. 1–1. DOI: 10.1109/MNET.2025.3551245 (To be Published)	✓	✓	
Workshop Demo	Chieh-Chun Chen, Navid Nikaein, and Mikel Irazabal Bengao. "Auto-RAN: Automated application-driven RAN slicing". In: *OpenAirInterface 10th Anniversary Workshop, 12-13 September 2024, Sophia Antipolis, France* (2024). URL: https://www.eurecom.fr/publication/7860	✓	✓	✓

Chapter 2

Flexible Control Plane

2.1 Introduction

The roll-out of 5G elevates user experience among distinct aspects and brings new opportunities to reshape mobile networks. Specifically, some key features, such as flexibility, openness, and intelligence, are expected to evolve monolithic infrastructures in legacy Radio Access Network (RAN). In this regard, standardization development organizations and industry fora, such as O-RAN Alliance [36] and Telecom Infra Project [84], are formed to not only concretize this vision into requirements, but also standardize open network interfaces among disaggregated RAN entities, such as Centralized Unit (CU), Distributed Unit (DU), and Radio Unit (RU).

This standardization has enabled the concept of "*Open RAN*", characterized by enhanced programmability and extensibility. A crucial aspect of this innovation is the adoption of SD-RAN, which leverages software-defined principles to enable dynamic and programmable control. In the following, we explore the role of the Control Plane (CP) in O-RAN and SD-RAN architectures, highlighting the current challenges and presenting the solutions proposed in this chapter.

2.1.1 Control Plane in O-RAN

In specific, within the O-RAN architecture [117], as illustrated in Figure 2.1, the RAN Intelligent Controllers (RICs) are central to providing software-defined control mechanisms, which are categorized into two types: Near-Real-Time (NearRT-RIC) [40] and Non-Real-Time (NonRT-RIC) [30]. The NonRT-RIC operates within the Service Management and Orchestration (SMO) framework and is designed to manage control loops with latencies greater than 1 s. Its key responsibilities include training and deploying Machine Learning (ML) and/or Artificial Intelligence (AI) models, managing policies (e.g., slice prioritization), and overseeing performance metrics (e.g., user and network performance).

Additionally, the NonRT-RIC communicates with the NearRT-RIC via the standardized A1 interface [29] to enforce policies and provide insights that facilitate near-real-time decision-making. Furhtermore, the NearRT-RIC handles control loops with latencies between 10 ms and 1 sec, focusing on near-real-time optimization of RAN-Nodes (e.g., CU and DU) over the E2 interface [42]. These RAN-Nodes are collectively referred to as E2-Nodes in the O-RAN terminology. The O1 interface [39], which serves the purpose of operations, administration, and management, is beyond the scope of this discussion.

To take it a step further, the Non-RT RIC and Near-RT RIC host two distinct types of control applications: rApps [30] for the NonRT-RIC and xApps [40] for the NearRT-RIC, as shown in Figure 2.1. The rApps provide value-added services for RAN operation and optimization (e.g., load balancing and energy savings) by communicating with the Non-RT RIC via the R1 interface [50]. In contrast, xApps communicate with network components through the NearRT-RIC Application Programming Interface (APIs), as illustrated by the light blue arrows in Figure 2.1, over various interfaces, encapsulating data into appropriate protocols. For instance, xApps interact with E2-Nodes via the E2 interface, with the NonRT-RIC via the A1 interface, and with the SMO via the O1 interface.

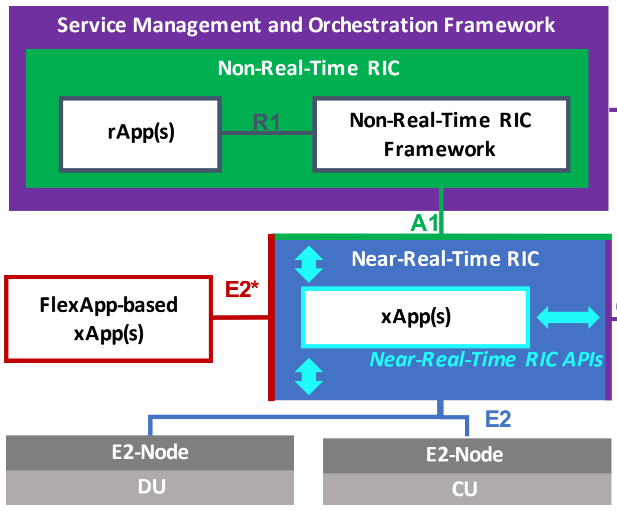


Figure 2.1: O-RAN architecture with the proposed FlexApp framework and E2* interface.

Specifically, to interact with RAN Functions (RFs) within E2-Nodes, such as the radio resource scheduler in the DU, xApps follow the data structures defined by standardized Service Models (SMs) [43] to send control actions or receive monitoring data[1]. These messages are generated through NearRT-RIC APIs (e.g., E2-related APIs [40]), encapsulated within E2SM [43], and transmitted to E2-Nodes over the E2 interface [42]. For example, the Key Performance Measurement (KPM) SM [41] is used to collect performance metrics (e.g., bit rate, error rate, and radio resource usage) from E2-Nodes and their connected User Equipments (UEs).

Challenges. Beyond the above recap, two key challenges remain. First, the interoperability of xApps across different NearRT-RIC platforms. Due to the lack of a standardized interface between the NearRT-RIC and xApps, current xApps rely on platform-specific APIs provided by the NearRT-RIC to interact with its functionalities. This reliance on proprietary APIs challenges the Open RAN vision of interoperability and openness, limiting xApp portability and multi-vendor integration. Second, the scalability of xApp deployments is constrained by the latency and overhead introduced by the NearRT-RIC. All messages sent to an xApp must first pass through the NearRT-RIC, where they are processed and exposed via its APIs. This centralized handling can become a bottleneck, especially for large-scale deployments or real-time RFs that require low-latency responses.

Contributions. To address these challenges, we propose the FlexApp framework not only provide interoperability for any native or third-party xApps, but also to enable ultra-low latency operations (<10 ms). This framework relies on our newly-designed interface - E2* - between NearRT-RIC and xApps (i.e., dark red lines in Fig. 2.1), which can be viewed as a natural extension of the E2 interface and is considered as a potential candidate for standardization. Further, this framework enables virtualization of RFs recursively to produce high-level abstraction and trigger future intent-based networking.

[1]SMs define the mechanisms for controlling and monitoring RAN functionalities in an E2-Node.

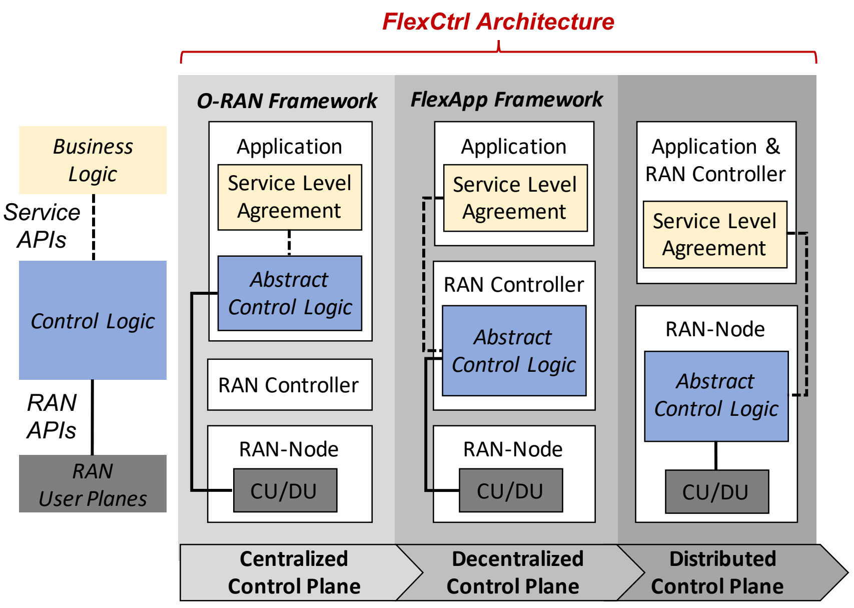


Figure 2.2: Hierarchical control of RAN user planes in SD-RAN (left) and three topologies for control plane in the proposed FlexCtrl architecture (right).

2.1.2 Control Plane in SD-RAN

Current SD-RAN platforms provide a programmable and centralized approach to managing RAN functionalities. A key feature of these platforms is the use of control applications (referred to as “App” in this context for clarity) running on top of the RAN controller, a concept also adopted by the O-RAN framework through constructs such as xApps operating on the NearRT-RIC, as illustrated in Figure 2.2. While this design facilitates dynamic management of RAN functionalities, it also brings significant challenges.

Challenges. One of the primary challenges lies in the centralized CP design. Managing multiple RAN-Nodes , through a single App on the RAN controller leads to increasing control complexity. This complexity grows as the RAN controller must coordinate multiple Apps, resolve potential conflicts, and ensure seamless interaction with the underlying RAN-Nodes. Furthermore, the lack of flexibility in supporting sub-10ms control loops limits the ability of current SD-RAN platforms to provide real-time control and monitoring, which are critical for achieving advanced functionalities such as adaptive RAN slicing and efficient resource optimization in highly dynamic network environments.

In particular, RAN slicing is a key feature of 5G-Advanced and future networks. It allows multiple services to run on the same infrastructure while ensuring performance guarantees and resource isolation. However, achieving effective RAN slicing requires a variety of standardized control options, further compounding the complexity of managing RAN User Planes (UPs). These challenges highlight the need for enhanced flexibility, scalability, and real-time control mechanisms in SD-RAN platforms to meet the evolving demands of next-generation networks.

To present a generalized approach and delve into the details of our proposed FlexCtrl architecture, three components are illustrated on the left side of Figure 2.2 (1) business logic, (2) control logic, and (3) RAN UPs. The business logic is defined by service providers in terms of the required Key Performance Indicators (KPIs), e.g., expected throughput and latency, in the Service Level Agreement (SLA). By invoking the exposed service APIs, business logic is converted into control logic, e.g., scheduling algorithms and parameters. Finally, the control logic is applied to RAN UPs, e.g., radio resource scheduler at the Medium Access Control (MAC) layer, by using the exposed RAN APIs.

From the viewpoint of service providers, the control logic can be centralized in the App, decentralized in the RAN controller, or distributed in RAN-Node(s), as shown in Figure 2.2. Their trade-off is mainly in terms of control loop latency and RAN-Node complexity. Take the centralized one (i.e., current O-RAN approach) as an example; the control logic is mostly located in the App; thus, it has the lowest RAN complexity but the highest control loop latency, which limits its real-time control capability. In contrast, decentralized and distributed topologies can not only reduce such control loop latency, but also enable App interoperability across various RAN controller platforms.

Contributions. In this chapter, we propose FlexCtrl, a flexible CP architecture that supports all three topologies (centralized, decentralized, and distributed) while offering both control flexibility through loosely coupled control logic locations and real-time programmability (down to microseconds). This architecture further enhances the proposed FlexApp and evolves the O-RAN architecture. Additionally, we redesign the radio resource scheduler, introducing a recursive scheduler for multi-level resource allocation, to demonstrate its capability in controlling RAN UPs.

2.2 Related Work

Software-Defined Networking (SDN) introduces a groundbreaking approach to networking by decoupling the Control Plane (CP) from the data plane, enabling programmable networks and centralized network management [101, 99, 115]. Using CP communication protocol like OpenFlow [111], SDN empowers network owners to directly program network devices, such as switches, through a centralized controller. This programmability, exposed through Application Programming Interfaces (APIs), supports dynamic management, efficient resource allocation, traffic optimization, as well as provides a global network view for intelligent decision-making and automated operations.

Historically, mobile networks have been opaque and tightly bound to vendor-specific hardware, creating closed ecosystems that are difficult to adapt or scale for rapidly evolving use cases and application demands [80]. However, with the increasing need for flexibility and scalability in next-generation infrastructures [148], mobile networks are being restructured around SDN principles, enabling programmability and vendor-agnostic architectures. Extending these principles to the RAN, Software-Defined Radio Access Network (SD-RAN) provides dynamic and scalable control, driving innovations across generations of mobile networks. In the following sections, we elaborate on the evolution of SD-RAN in 4G, 5G, and beyond, highlighting key advancements and challenges.

2.2.1 SD-RAN Platforms in 4G Network

One such innovation is SoftRAN [85], which introduces the concept of SD-RAN with a centralized CP topology by abstracting all base stations within a local geographical area into a single "big base station." This abstraction is composed of: (1) a central controller, which acts as the centralized CP, maintaining a global network view to manage time-flexible and coordination-critical tasks (e.g., centralized handovers); and (2) radio elements, which form the distributed CP, handling time-sensitive, localized control decisions (e.g., radio resource scheduling) at individual base stations. While SoftRAN offers control flexibility by combining centralized and distributed CPs, it has limitations in scalability across multiple Radio Access Technologies (RATs) and relies on vendor-specific southbound APIs to interface with underlying technologies.

Building upon this need for scalability and interoperability, 5G-EmPOWER [77] offers a RAT-agnostic architecture that enables a centralized controller (referred to as the operating system in [77]) to manage various RATs, such as LTE, NR, and Wi-Fi, through the southbound interface using its OpenEmpower protocol. This architecture abstracts the complexity of underlying radio technologies, allowing network operators to access the network state and perform control actions through management applications that communicate with the centralized controller via the northbound interface using high-level Domain-Specific Languages (DSLs). However, the centralized CP design introduces a fixed control loop latency, which may pose challenges for latency-sensitive decision-making, such as radio resource scheduling operating within millisecond or sub-millisecond intervals.

In parallel, FlexRAN [82] takes a hybrid approach, combining centralized and decentralized CP architectures. It achieves this by virtualizing the complex control logic of the RAN into modular functions within a centralized controller. This controller interacts with the RAN infrastructure through a custom-tailored southbound API, supporting programmability and enabling operators to develop control applications (App) through its northbound API. FlexRAN also allows dynamic modification of virtualized control functions, facilitating flexible control delegation, monitoring, and management of RAN operations. However, its northbound API exposes lower-level details rather than offering high-level abstractions, making application development more complex. Moreover, while the customizable southbound API enhances flexibility, it limits the controller's ability to seamlessly control multiple RANs and RATs from different vendors, reducing its overall scalability and interoperability.

Adding another dimension to this landscape, Self-Organizing Networks (SON) [26, 52], focus on automating network management tasks. SON enables dynamic configuration, handover control, interference management, and load balancing, making networks more adaptable to fluctuating conditions. Despite its programmability and automation capabilities, SON implementations have historically relied on vendor-specific solutions, which restrict their flexibility and hinder full interoperability across diverse network environments.

These innovations collectively highlight the potential benefits of SD-RAN, yet they remained largely experimental and in their infancy during the 4G era, with limited real-world deployment. Furthermore, they underscore persistent challenges, such as interoperability, latency, and dependence on vendor-specific implementations that next-generation architectures might overcome. In 5G, SD-RAN principles are increasingly adopted, with Open RAN, led by the O-RAN Alliance [36], making significant progress in addressing these challenges, as detailed in the next section.

2.2.2 SD-RAN Platforms in 5G Network

Building on the foundational ideas of SD-RAN, O-RAN introduces a groundbreaking approach to 5G networks by disaggregating hardware and software, enabling vendor-agnostic interoperability through standardized interfaces (e.g., E2 [42]), as well as enabling programmability through AI/ML-based solutions. These advancements mark a transition toward more open (vendor-agnostic), scalable, and programmable mobile network architectures, paving the way for further innovation in 5G and beyond [137]. Research leveraging the O-RAN architecture [117] demonstrates its potential to enhance 5G networks and address these challenges.

FlexRIC [134] is an open-source Near Real-Time RAN Intelligent Controller (NearRT-RIC) platform compliant with O-RAN standards that implements the standardized E2 interface protocol to enable seamless communication between RAN nodes and controller. This standardized control protocol ensures interoperability across vendors and RATs, addressing key challenges from the 4G era (see Section 2.2.1). Leveraging the centralized CP architecture defined by O-RAN, FlexRIC enables network operators and vendors to develop App, known as xApps in O-RAN terminology, which control and monitor RAN operations using standardized Service Models (SMs).

Unlike the NearRT-RIC implementation provided by the O-RAN Software Community (OSC) [119], which faces practical limitations from bugs [95], unnecessary feature overhead [134], and insufficient support for emerging use cases cases, FlexRIC addresses these shortcomings by offering a more robust, flexible, and extensible platform. By enabling vendors to extend and customize SMs, FlexRIC not only supports the creation of specialized SD-RAN controllers but also enables advanced capabilities such as fine-grained traffic control [90] and dynamic radio resource slicing tailored to specific QoS requirements [134], going beyond the scope of standardized SMs. The importance of customizable SMs is further demonstrated in works like [95, 97], where tailored SMs were developed for use cases such as programmable network slice configuration and statistics collection. These examples highlight the growing need to extend O-RAN standardized SMs and enable their customization to meet the demands of emerging use cases and applications.

Additionally, project JANUS [83] underscores the limitations of standardized SMs, which evolve slowly and do not scale efficiently. Every new use case requires a new SM, forcing collaboration with a specific RIC and RAN vendor to add support for each SM. JANUS addresses this limitation by introducing dynamic virtual RAN functionality, extending the RIC to allow operators and trusted third parties to write and deploy their own telemetry, control, and inference logic (referred to as codelets) at runtime. However, it deviates from

O-RAN specifications, which may limit interoperability across different RAN vendors. In contrast, FlexRIC [134] not only offers enhanced customizability but also builds on O-RAN specifications, inheriting its core interoperability features.

Despite these advancements, mobile networks still face significant barriers in achieving comprehensive control flexibility across the entire system. A major challenge is that current approaches [83, 96, 134] primarily focus on enhancing the programmability of RAN functionalities while the flexible control of Apps and controllers within the SD-RAN architecture remains underdeveloped. For instance, each use case may require one or more Apps, such as data monitoring, slice control, or handover control. This brings up critical questions for operators, such as which Apps should be selected and deployed in the network. At the same time, vendors face the challenge of developing Apps and controllers that can be dynamically managed to adapt to fluctuating network conditions while responding to control decisions made by operators. Addressing these challenges is essential for enabling adaptive and intelligent network control in SD-RAN.

Another challenge is the limited usability of Apps, which are often constrained by platform-specific implementations. For instance, xApps communicate with the NearRT-RIC via E2-related APIs [40], which are not standardized as open interfaces - E2 [42]. This lack of standardization frequently results in vendor lock-in, making it difficult to port Apps across different controllers or platforms. In addition, the development complexity of Apps continues to grow. This is due not only to the intricate data structures defined in standardized SMs but also to the increasingly sophisticated control logic required to manage heterogeneous RAN environments.

2.3 Why Flexible Control Plane?

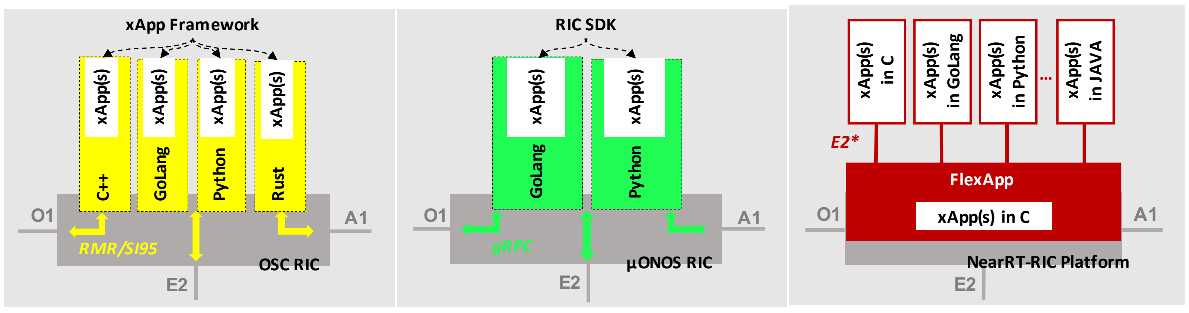
In this section, we discuss the need for a flexible CP design in the context of O-RAN and SD-RAN, respectively. We begin with the FlexApp framework, which enables platform-agnostic xApps within the O-RAN architecture. Next, we introduce FlexCtrl, which builds on the FlexApp framework to deliver low-latency and customized control operations. Finally, we highlight the role of FlexCtrl in controlling heterogeneous RAN and enabling advanced use cases like RAN slicing.

2.3.1 Platform-Agnostic xApp in O-RAN

To avoid lock-in with a closed and/or proprietary xApp framework, we compare FlexApp framework with two others proposed by the open-source community: one developed by the O-RAN Software Community (OSC) and the other by the Open Networking Foundation (ONF). While the OSC and ONF xApp frameworks are deployed on their respective NearRT-RIC platforms, the FlexApp framework offers a distinct approach, as shown in Figure 2.3 and detailed in Table 2.1.

Table 2.1: State-of-the-art comparison for open-source xApp framework.

xApp Design Approach	**Open-Source xApp Frameworks**		
	OSC [123]	**ONF** [124]	**FlexApp**
Operating Platform	OSC RIC	ONF µONOS RIC	Any RIC
Communication Interface	RMR/IS95 library	gRPC-based APIs	E2*
Baseline SDK	Language-specific	Language-specific	Language-agnostic
Programming Language	C++, Golang, Python, Rust	Golang, Python	Supports a broad range of languages
Deployment on NearRT-RIC	Embedded	Embedded	Isolated
Complexity of Data Composition Across Multi-Vendor/RAT E2-Nodes	High	High	Low, with simplified data recursively exposed at the northbound interface



(a) OSC xApp framework (b) ONF RIC SDK (c) FlexApp framework

Figure 2.3: Comparison of xApp framework from OSC, ONF and the proposed FlexApp.

OSC xApp Framework. The OSC presents its xApp framework [123] (see Figure 2.3a) on its own NearRT-RIC platform (OSC RIC [119]) and introduces the RIC Message Router (RMR)[2] to communicate with the xApps through its Software Development Kit (SDK). The RMR is a library that abstracts the message transport mechanism (e.g., Nanomsg, Nanomsg next generation, or Socket Interface-95 [SI95] [120]) to be used by xApps to send/receive messages to/from an E2-Node. The message routing and endpoint selection in RMR are based on a pair of message type and subscription ID, which need to be translated into the endpoint via a table entry. A non-negligible overhead and latency are observable when this framework is employed due to the process of selecting an endpoint for each message [121].

ONF xApp Framework. Moreover, the ONF introduces its xApp framework, referred to as RIC SDKs in Figure 2.3b, for developing xApps on its own NearRT-RIC platform (µONOS RIC) [124]. These xApps communicate with the µONOS RIC using gRPC-based APIs[3], which provide HTTP request/response communication. Such an SDK encapsulates some of the complexities of dealing with individual gRPC services (e.g., threading and session management) to offer routing capabilities, and it requires xApps to provide the endpoint address (e.g., DNS server hostname, IP address, and port number) [124]. Despite claims that gRPC is faster than other REST-like transaction implementations, when compared to the RMR on top of SI95 (RMR/SI95) [122], gRPC cannot achieve the same throughput and with only acceptable latency when the message rate is less than 10,000 per second.

[2]https://github.com/o-ran-sc/ric-plt-lib-rmr
[3]https://github.com/onosproject/onos-api

FlexApp Framework. In contrast, the proposed FlexApp framework (see Figure 2.3c) extends the standardized E2AP protocol as an E2* interface. This new interface can onboard the xApps into the NearRT-RIC and supports bidirectional communication between xApps and NearRT-RIC while inheriting the existing mechanism in the E2, i.e., Stream Control Transmission Protocol (SCTP) socket. One key aspect of this framework is to avoid platform lock-in; hence, the NearRT-RIC platform in Figure 2.3c can be any O-RAN compliant RIC (e.g., OSC RIC in Figure 2.3a, µONOS RIC in Figure 2.3b, or FlexRIC [134]). This framework also supports xApp development in a variety of high-level programming languages, such as Python, GoLang, and Java.

Another advantage of the proposed FlexApp framework is its ability to enable xApps to specialize in a desired set of SMs, effectively decoupling the SMs from the NearRT-RIC. In the current approach, different formats for a given SM are standardized, allowing xApps to manage the SMs' internal data models by subscribing to specific events and actions. Consequently, an xApp must compose the desired SMs from different E2-Nodes and aggregate the data accordingly which increases the complexity of managing data in an xApp, particularly as the number of E2 Nodes grows. In contrast, the FlexApp framework simplifies this process by enabling the composability of data collected from various SMs at an intermediate level. It achieves this by using a virtualized RAN Function (vRF) to consolidate the data from different E2-Nodes, which is then exposed to higher-level xApps through its recursive northbound interface.

2.3.2 Low-Latency and Customized Control Operations

The O-RAN architecture introduces two operational cases for the RIC, based on control loop durations. Tasks sensitive to latency, such as radio bearer, radio resource, interference, and mobility managements, are assigned to the NearRT-RIC, which operates with control loops under 1 s [40]. Meanwhile, tasks requiring longer processing times for large-scale data analysis and optimization, such as energy efficiency, predictive traffic steering, and long-term network capacity planning using AI/ML models, are handled by the NonRT-RIC, which supports control loops exceeding 1 s [30].

While the O-RAN approach supports these two cases, certain advanced RAN functionalities, such as beamforming and Time Division Duplex (TDD) managements, as well as Quality of Service (QoS) adjustments, often require control loop intervals in the sub-10 ms or sub-millisecond range, demanding ultra-low-latency control responses under dynamic network conditions. These stringent requirements cannot be met by the NearRT-RIC or NonRT-RIC due to their longer control loop intervals. Consequently, real-time systems, such as DU and RU, are necessary for immediate, direct control. Additionally, the O-RAN architecture lacks sufficient flexibility to adapt the placement of control logic based on specific deployment needs, as its centralized applications (e.g., xApps and rApps) limit optimization for latency requirements.

For instance, use case like adaptive RAN slicing across multiple E2-Nodes, which dynamically allocates radio resources to delay-critical services, cannot be effectively supported under the current O-RAN architecture. Such use case rely on real-time control

Table 2.2: State-of-the-art comparison for control plane design in SD-RAN.

Control Plane Design	Control Logic Placements(s)	Support Control Plane Topologies			Minimum Control Latency	Average RAN Complexity
		Centralized	Decentralized	Distributed		
SoftRAN [85]	RAN Controller, RAN-Node	✓	-	✓	Low	Low
FlexRAN [82]	App	✓	-	-	High	Low
O-RAN [117]	App	✓	-	-	High	Low
HexRAN [96]	App	✓	-	-	High	Low
RAN Engine [135]	RAN-Node	-	-	✓	Low	High
dApp [81]	RAN-Node	-	-	✓	Low	High
JANUS[83]	RAN-Node	-	-	✓	Low	High
FlexApp	**App, RAN Controller**	✓	✓	-	**Medium**	**Low**
FlexCtrl	**App, RAN Controller, RAN-Node**	✓	✓	✓	**Low**	**Low**

mechanisms and close coordination between E2-Nodes to ensure consistent performance and meet stringent SLAs. Depending on the specific requirements, control logic may need to operate closer to the E2-Nodes (e.g., real-time scheduling decisions), moderately farther from them (e.g., interference management), or significantly farther from them (e.g., energy management). These limitations underscore the need to enhance the O-RAN architecture by distributing CP in real-time systems, supporting ultra-low-latency control operations, and enabling flexible control logic placement for diverse deployment scenarios.

The proposed FlexCtrl architecture addresses these challenges by supporting all three CP topologies: centralized, decentralized, and distributed, as shown in Figure 2.2. This flexibility allows for a balance between control latency and RAN complexity, making it adaptable to diverse deployment scenarios. Unlike existing solutions that rely on specific CP topologies and are limited to particular use cases, FlexCtrl provides a versatile and unified approach. A detailed comparison of the proposed solutions, comprising the FlexCtrl architecture and the FlexApp framework, against the current O-RAN approach and related works is presented in Table 2.2.

The centralized CP topology [85, 82, 117, 96] simplifies RAN implementations by centralizing control logic within App(s) or the RAN controller. However, it introduces additional control latency due to the need to translate business logic into control logic for heterogeneous RANs. This translation further increases the complexity of App(s) or RAN controller and exacerbates control delays, making it less suitable for scenarios handling low-latency services.

To address these latency challenges, the distributed CP topology [135, 81, 83] embeds control logic directly within the RAN-Nodes. This approach eliminates intermediate processing layers, enabling ultra-low-latency control operations. However, while it minimizes latency, it significantly increases the complexity of RAN implementations. Each RAN-Node must independently handle sophisticated control tasks, which complicates coordination across multiple RAN-Nodes and makes the system harder to manage at scale.

In contrast, the decentralized CP, as implemented in the FlexApp framework, strikes a balance by bringing control logic closer to the RAN-Nodes while leveraging an abstraction layer within the RAN controller. This abstraction layer efficiently manages business logic and applies it across multiple RAN-Nodes, reducing control latency without the full complexity of a distributed topology. Additionally, FlexApp simplifies App(s) by offloading complex processing to a decentralized yet abstracted layer, while retaining the capability to coordinate multiple RAN-Nodes effectively.

Building on these concepts, the FlexCtrl architecture balances the trade-offs of all three CP topologies. It enables operators to adaptively allocate control logic to centralized, decentralized, or distributed layers based on specific deployment requirements. By providing this flexibility, FlexCtrl ensures low-latency control operations while maintaining manageable RAN complexity, making it a robust and adaptable solution for dynamic network conditions.

2.3.3 Controlling Heterogeneous RAN

Controlling heterogeneous RAN becomes increasingly complex when managing diverse technologies across multiple RAN-Nodes from different vendors. A flexible CP serves as the glue that unifies and optimizes such environments, providing seamless management of RAN-Nodes regardless of vendor-specific differences. The FlexApp framework introduces abstraction for control logic, while the FlexCtrl architecture provides three CP topologies to support diverse deployment needs. Together, they enable efficient resource management, low-latency control operations, and advanced use cases such as RAN slicing, making it possible to support multi-service and multi-tenancy in heterogeneous RAN.

In the context of RAN slicing, the radio resource scheduling aims to use limited spectrum for distinct criteria, e.g., Proportional Fairness (PF). Several works explore further performance metrics, like the ones introduced in NVS [100] and Earliest Deadline First (EDF) [86] algorithms, by taking particular service requirements into account. Another aspect is how radio resources are allocated, e.g., the work in [102] provides a two-level radio resource scheduler (slice- and user-level) by abstracting radio resources from Physical Resource Blocks (PRBs) to virtual RBs (vRBs).

Building on these ideas, our objective is to increase the modularity and extensibility of the radio resource scheduler, so that the control logic in the FlexCtrl architecture can flexibly construct a multi-level scheduling function. To achieve it, we redesign the MAC scheduler recursively in Section 2.5.

2.4 FlexApp Framework

In this section, we provide an overview of the FlexApp framework within the context of the O-RAN architecture, focusing on how it enhances the scalability and reusability of control applications. We also delve into the design details of FlexApp, emphasizing its core principles and key capabilities.

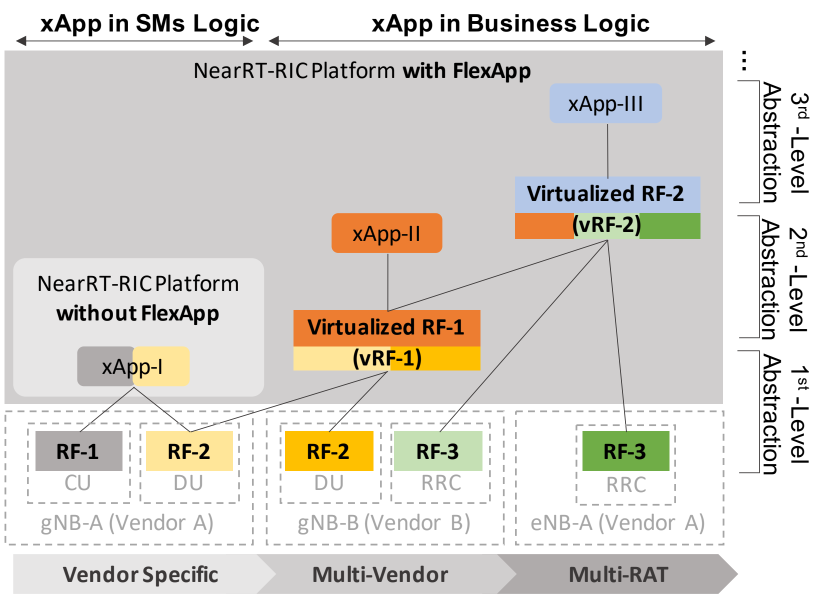


Figure 2.4: NearRT-RIC platform without and with the proposed FlexApp framework and three xApp examples (xApp-I, xApp-II, xApp-III).

2.4.1 Overview

As discussed in Section 2.1, the FlexApp framework is proposed to address challenges such as platform lock-in and limited xApp reusability. A key approach adopted in FlexApp is to offer a novel set of abstractions for the vRFs to provide a high level of SM compatibility and composability, as depicted in Figure 2.4. In fact, this approach can distinguish the xApp development model by relying on SMs' logic from that relying on business logic [40]. To provide more insight, three cases are presented in the same figure.

The first legacy case is xApp-I in Figure 2.4, in which this xApp needs to handle different RFs (i.e., RF-1 and RF-2) through the corresponding SMs, data, and control actions and form a virtualized base station to process the data, even in a vendor-specific case. In contrast, the FlexApp framework provides the notion of vRFs to enable flexible SM composition based on the xApp requirements. Therefore, vRF-I in Figure 2.4 is formed, which reveals a single SM abstracting data from the 2 RFs from different vendors to the xApp-II. Further, a higher level of abstraction can be built via recursive vRFs referencing, e.g., vRF-2 in Figure 2.4 can handle the data from both RF-3 and vRF-1 among different Radio Access Technologies (RATs) to serve the purpose of xApp-III. Such virtualization allows the construction of new vRFs from existing RFs/vRFs as well as the recomposition of SMs, while still complying with E2 interface requirements.

As a concrete example, the vRF related to Quality of Service (QoS) management can be composed by virtualizing other RFs that handle the data rate (e.g., by scheduling resource blocks) and latency (e.g., by managing queues). Furthermore, this vRF can also virtualize another vRF that handles service-specific metrics, such as RAN availability, according to the business logic in xApp.

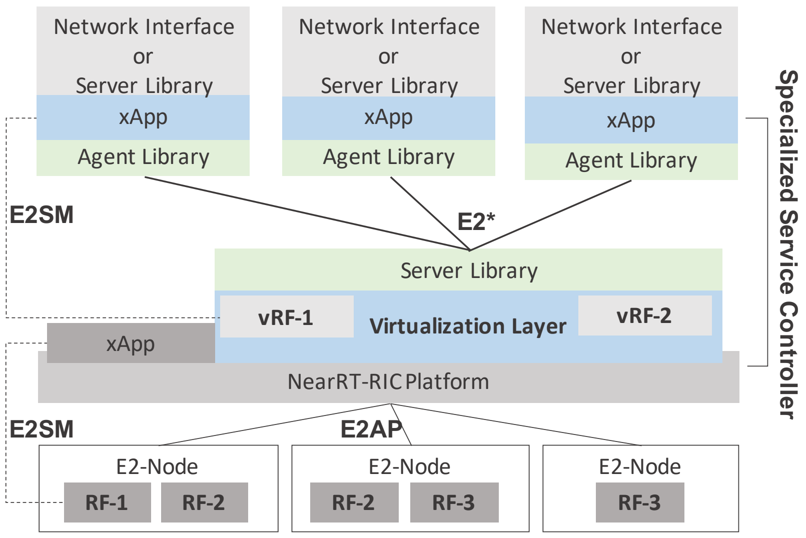


Figure 2.5: The design of FlexApp framework.

To conclude, our proposed FlexApp framework can benefit both RAN providers and xApp developers. On one hand, it simplifies the development of RFs and lowers the overhead on the E2-Nodes, since more advanced vRFs become composable via multi-level abstractions. On the other hand, it abstracts the heterogeneous RAN deployment and allows the newly composed vRF to be tailored to the desired xApp objectives, thereby simplifying the development of xApp. Furthermore, unlike the relationship between NearRT-RIC and xApps using platform-specific NearRT-RIC APIs (cf. Figure 2.3a and Figure 2.3b), xApps under the FlexApp framework are independent components. They have independent lifecycles and are treated as "*first-class citizen applications*" with a dedicated E2* interface to transfer the E2-based context for compatibility.

2.4.2 Design Details

Generally, the FlexApp framework follows the zero-overhead principle to achieve low latency, and it includes four main elements, as shown in Figure 2.5: (1) E2* interface, (2) server library, (3) virtualization layer, and (4) agent library. From the viewpoint of the xApp, a *specialized service controller* is formed by utilizing these elements to realize its control logic. We can see a depiction of such a controller in Figure 2.6, and the design details of these elements are given as follows:

E2* Interface. The E2* interface supports elementary procedures (e.g., E2* setup request and response) and services (e.g., report and control) similar to the E2AP protocol [42]. First, like the E2 interface, the E2* setup request message is initiated by the agent towards the NearRT-RIC, and, once successfully received, the NearRT-RIC assigns a particular xApp ID and responds with the information of connected E2-Nodes. However, unlike E2, the E2* subscription request message is generated by the agent to the NearRT-RIC, including the E2-Node(s) that this xApp would like to subscribe to. This message is bookkept (e.g., up-

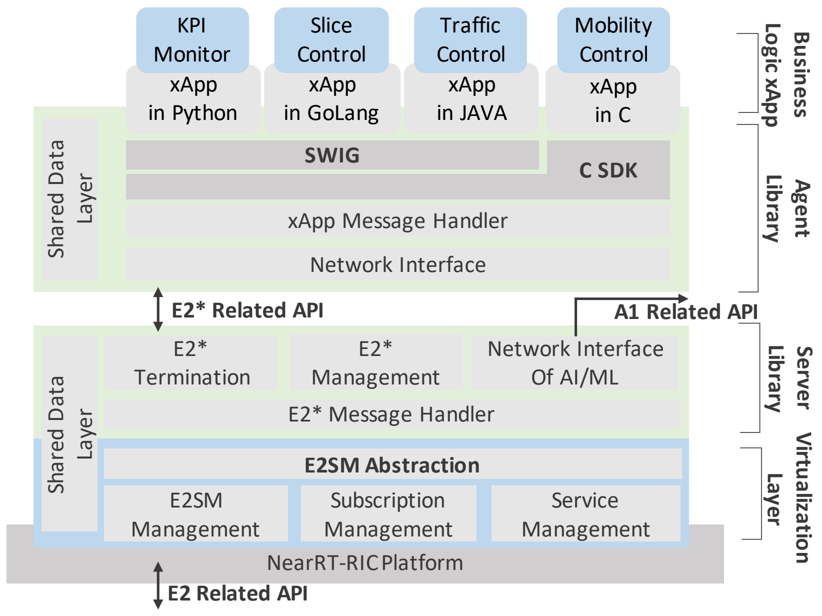


Figure 2.6: A specialized service controller based on the FlexApp framework.

date the subscription request ID) and then forwarded to the corresponding E2-Node by the NearRT-RIC. A similar process is followed by a control request message. In total, four new elementary procedures are added to the E2 ASN.1 specification: E2* Setup, RIC Subscription, RIC Subscription Delete, and RIC control. Note that other encoding and decoding schemes (compared to the standard ASN.1) are supported in the E2* interface using *C11 _Generics*, which can alleviate existing bottlenecks at the CPU.

Server Library. The server library is essential for extending the NearRT-RIC with a virtualization layer that can aggregate vRFs from RFs/vRFs, as mentioned in Section 2.4. Moreover, it manages the connection towards the agents and multiplexes messages between the virtualization layer and the agents by using the E2* termination, E2* management, and E2* message handler modules in Figure 2.6. Also, this library is designed as an event-driven/callback-driven system, adhering to the aforementioned ultra-lean design principle to impose minimal overhead. Therefore, it activates the virtualization layer only when there are new messages to be handled.

Virtualization Layer. The virtualization layer can implement new SMs on top of the newly composed vRF and expose specific information to xApps. From the xApp viewpoint, this layer is the key in the specialized service controller, and there are four modules in this layer (cf. Figure 2.6). First, the E2SM abstraction handles the vRFs and the respective SMs to expose the prepossessed information to the northbound xApp via the E2* interface. Moreover, the E2SM management module handles the information of the supported SMs in the NearRT-RIC. Then, the subscription management module is responsible for the subscription procedure and tracks the existing RF subscription statuses for each xApp. Finally, the service management module takes care of the interaction between xApps and (v)RFs by processing the messages related to the basic services specified in E2SM, including report, insert, control, and policy.more than one xApps sends conflicting messages to the same (v)RFs.

Agent Library and xApp Development. From the perspective of xApp, the agent library is the cornerstone of realizing its logic because it is built on top of this library in the southbound. Furthermore, as shown in Figure 2.5, an xApp can recursively expand the interface at its northbound by reusing the server library. In detail, two building blocks exist in this library as a part of the SDKs: C SDK and SWIG[4] as shown in Figure 2.6. The C SDK refactors all common functionalities of an xApp (e.g., communication, message handling, encoding/decoding, and data layer) under a single C library to be exposed to xApps for the development of C/C++ based xApp. In contrast, as mentioned in Section 2.3, to enable the xApp development using other high-level programming languages, the SWIG compiler is apoted to create the wrapper codes for the C-based library of E2/E2* interfaces. To develop an xApp, the northbound SDK is provided in the agent library functions including:

- init_xapp_api: Initializes the xApp by sending E2* setup request to the NearRT-RIC.
- e2_nodes_xapp_api: Retrieves information about connected E2-Nodes.
- report_sm_xapp_api: Enables report service by subscribing to a RF in an E2-Node.
- control_sm_xapp_api: Controls a targeted RF on an E2-Node..

In short, the FlexApp framework provides several key takeaways: (a) Forward compatibility, such as dedicated controllers and xApps, (b) Ultra-lean design by extending E2AP without extra overhead, (c) A unified SDK exposing generic APIs for xApp development, and (d) Multilingual xApp development kit.

2.4.3 Implementation and Testbed Setup

We prototype the proposed FlexApp framework by implementing it on the top of FlexRIC [134], which is the O-RAN compliant NearRT-RIC platform developed by the OpenAirInterface (OAI) [114]. This implementation validates the performance of the E2* interface (referred to as the E42 interface in the FlexRIC implementation) and the concept of the virtualization layer in the NearRT-RIC, as detailed in the evaluation results presented in Section 2.6.

Furthermore, we setup two testbeds to evaluate the FlexApp framework: a containerized (cloud-native) environment and a non-containerized (bare-metal) environment. The cloud-native testbed runs on Ubuntu 20.04.5, with 20 CPUs at 4.8 GHz, 64 GiB of memory, and uses Kubernetes v1.24.5 alongside Containerd v1.6.8. The bare-metal testbed operates on Ubuntu 20.04.1, with 12 CPUs at 4.7 GHz and 32 GiB of memory, without any container orchestration.

To represent the functionalities within RF and perform the RIC report service [40], we use the E2-emulator provided by the FlexRIC. Within each E2-emulator, the Medium Access

[4]https://www.swig.org/

Control (MAC) RF is enabled, and the corresponding data is encapsulated into a customized SM - MAC SM. Then, our self-developed xApp subscribes to the MAC SM with a particular periodicity, and the composed E2SM will be periodically updated by the E2-emulator using E2AP and sent to the xApp using the E2* interface.

2.5 FlexCtrl Architecture

In this section, we provide an overview of the FlexCtrl architecture, which extends the FlexApp framework, and compare its three control topologies, using RAN slicing as a representative use case. Additionally, we delve into the design details of FlexCtrl, focusing on the abstraction of control logic and the implementation of the redesigned radio resource scheduler.

2.5.1 Overview

Building on the above foundation, we introduce the FlexCtrl architecture, which integrates the FlexApp framework and extends the current O-RAN architecture. This extension enables the flexible deployment of control logic across centralized, decentralized, and distributed topologies (see Figure 2.2), providing the adaptability needed to meet diverse deployment scenarios and achieve shorter control loop latencies.

One use case that demonstrates this flexibility is adaptive RAN slicing, where the radio resource scheduling function is implemented as a recursive design, allowing heterogeneous control actions to be performed based on the control logic placement. Moreover, the recursive design concept can be extended to other RFs, empowering RAN-Nodes to efficiently support multi-service and multi-tenancy operations.

In Figure 2.7, a high-level architecture of FlexCtrl is shown, containing business logic (yellow), control logic (blue), and RAN UPs (gray). The service provider defines the business logic[5], which specifies the required KPIs for a particular service, such as throughput and latency, as part of the SLA. These KPIs are represented in Figure 2.7 as ExpTput and ExpLat, respectively. Control logic is then responsible for meeting these SLAs by instantiating appropriate control actions (e.g., create network slices) on the controllable RFs within the RAN UP (e.g., radio resource scheduling functions). Depending on the topology used (cf. Figure 2.2), control logic can reside in the App, the RAN controller, or the RAN-Node:

- **Centralized**: Apps handle business logic and control logic. As shown in Figure 2.7, App-I abstracts the expected throughput of Service 1 into control logic, e.g., PRB number, in a comprehensible manner to the radio resource scheduling function (denoted as SCH in Figure 2.7) within RAN-Node A.
- **Decentralized**: Apps define business logic and use service APIs exposed by the RAN controller. In Figure 2.7, App-II and App-III request the expected throughput for

[5]Business logic is used to encode real-world business rules as a sequence of commands or actions, and can be defined in a nested way.

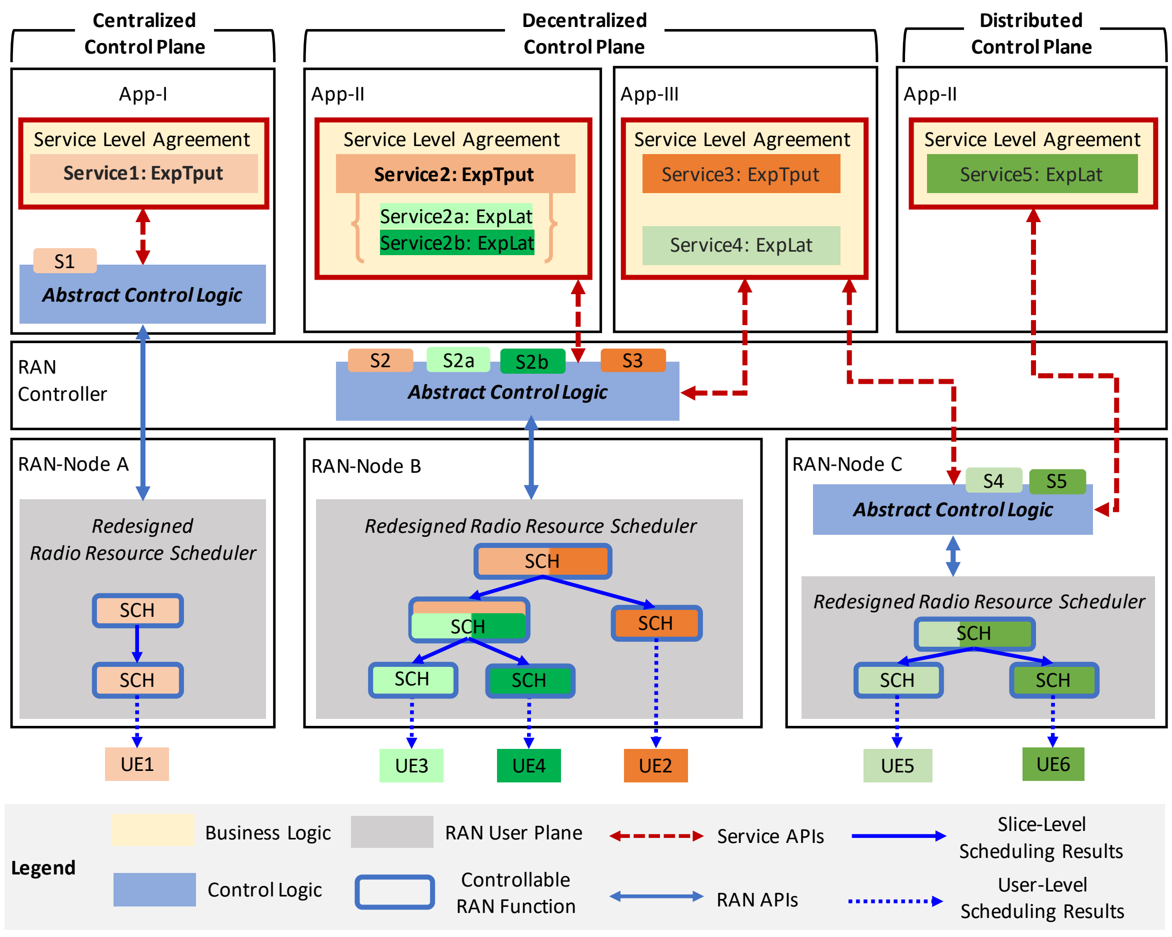


Figure 2.7: FlexCtrl architecture with three abstract control logic examples.

Services 2 and 3, respectively, and Service 2 asks for extra expected latency for its two sub-services (2a and 2b). The RAN controller abstracts all these requests into the control logic and uses RAN APIs toward SCHs within RAN-Node B.

- **Distributed**: RAN-Nodes incorporate both controllable functions in RAN UP and control logic, allowing shorter control loops in real-time. Consider App-III and App-IV in Figure 2.7, RAN-Node C embeds the abstracted control logic from the expected latency for Services 4 and 5 directly into the SCHs within RAN UP.

Moreover, the abstract control logic shown in Figure 2.7 unifies the underlying functions within RAN UP and used SMs across various dimensions, including multi-vendor and multi-RAT RAN-Nodes. Thus, it allows tailoring the control logic to the service requirements in terms of the requested KPIs. Such an “abstraction layer” is essential to FlexCtrl.

Entering the controllable RFs within RAN UP as depicted in Figure 2.7, three redesigned radio resource scheduler are shown in RAN-Nodes A, B, and C for Services 1, Services 2 and 3, as well as Service 4 and 5, respectively. These RFs are realized by employing the recursive approach and will be detailed in the next subsection, which allows for multi-level scheduling. At the slice level, radio resources are partitioned based on (sub-)slice information (e.g., SLA). These per-slice resources are then allocated to the associated UEs

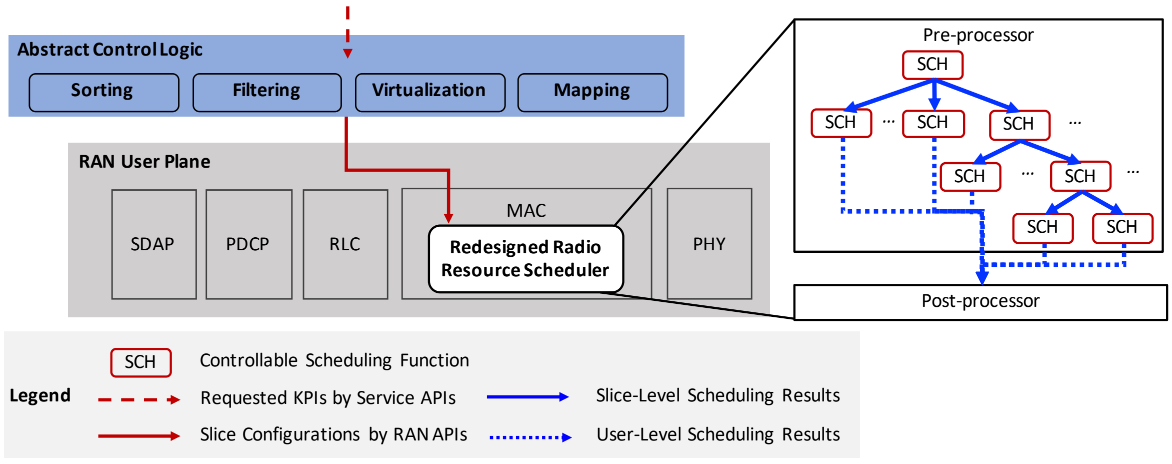

Abstract Control Logic
Sorting
Filtering
Virtualization
Mapping
RAN User Plane
SDAP
PDCP
RLC
MAC
Redesigned Radio Resource Scheduler
PHY
Pre-processor
SCH
Post-processor
Legend
SCH Controllable Scheduling Function
Requested KPIs by Service APIs
Slice Configurations by RAN APIs
Slice-Level Scheduling Results
User-Level Scheduling Results

Figure 2.8: Abstract control logic and redesigned radio resource scheduler for recursive operation in the RAN slicing use case.

based on UE information, e.g., QoS. In Figure 2.7, the colors within each SCH represent the radio resources allocated for the corresponding service.

2.5.2 Design Details

In this section, we elaborate on two key enablers of FlexCtrl: (a) abstract control logic, and (b) redesigned radio resource scheduler to support recursive operations in the RAN slicing use case as an example.

Abstract Control Logic. The aim of abstract control logic is to allow network operators to define the characteristics of each slice in a flexible manner based on the KPIs of each service without disclosing specific deployment details (e.g., multi-vendor and multi-RAT at various sites). As depicted in Figure 2.8, the process of controlling RAN slices is done by the "mix-and-match" abstract control logic between the requested KPIs for each service and the respective slice configuration (e.g., radio resource share), which includes setting up slices, and deciding the scheduling algorithm and parameters. Specifically, four tasks are included as shown in Figure 2.8:

1. **Sorting**: Arrange services based on their requested KPIs or non-technical criteria,
2. **Virtualization**: Virtualize radio resources of RAN-Nodes with tailored approach (e.g. PRB to vRB),
3. **Filtering**: Adjust the requested KPIs in accordance with the (virtualized) resources of RAN-Nodes,
4. **Mapping**: Create slice configurations by mapping the requested KPIs to use respective RFs (e.g., scheduling algorithm and parameters).

Redesigned Radio Resource Scheduler. RFs redesign is key to FlexCtrl, and we focus on the radio resource scheduler as an example. First, it is divided into two parts in Figure 2.8: (a) pre-processor and (b) post-processor. The former incorporates the radio resource scheduling function (denoted as SCH in Figure 2.8) and can be controlled as a RF, while the latter allocates resources to physical channels based on the results from the pre-processor. Then, the SCH is redesigned recursively to dynamically apply different algorithms and parameters based on performance requirements and scheduling constraints, effectively realizing both slice- and user-level scheduling. By doing so, the original scheduling problem can be decomposed, further increasing flexibility and extensibility.

Algorithm 1 Recursive radio resource scheduling

```
    Global Variable
    $\mathbf{C}$ is a vector of size $\mathcal{C}$ with slice configuration (scheduling algorithm and associated
    user information).
    $\mathbf{S}_i$ is a vector containing the number of scheduled resources to $i$-th slice slice and
    its associated UEs.
    Input
    $l$ is the scheduling level, equal to $slice$ or $user$.
    $r$ is the number of available resource blocks.
    $\mathcal{N}$ is the set of unscheduled slices.
 1: procedure SCH($l, r, \mathcal{N}$)
 2:     if $r > 0$ then
 3:         if $l = slice$ then
 4:             $i \leftarrow SelectSlice(\mathbf{C}, \mathcal{N})$
 5:             $\mathbf{S}_i[0] \leftarrow \min(r, AllocRB(r, \mathbf{C}[i], 0))$ ;
 6:             $r \leftarrow r - \mathbf{S}_i[0]$;
 7:             $\mathcal{N} \leftarrow \mathcal{N} \setminus \{c_i\}$;
 8:             if $length(\mathbf{S}_i) > 1$ then $SCH(user, \mathbf{S}_i[0], \mathcal{N})$
 9:             end if
10:             if $|\mathcal{N}| > 0$ then $SCH(slice, r, \mathcal{N})$;
11:             end if
12:         else
13:             $j \leftarrow 1$
14:             while $r > 0$ and $j < length(\mathbf{S}_i)$ do
15:                 $\mathbf{S}_i[j] \leftarrow \min(r, AllocRB(r, \mathbf{C}[i], j);)$
16:                 $r \leftarrow r - \mathbf{S}_i[j]$;
17:                 $j \leftarrow j + 1$;
18:             end while
19:         end if
20:     end if
21: end procedure
```

Practically, this recursive approach is provided in details in Algorithm 1, and we first define two sets:

- $\mathcal{C} := \{c_1, \cdots, c_{|\mathcal{C}|}\}$ contains all slices to be scheduled.

- $\mathcal{U}_i := \left\{u_1, \cdots, u_{|\mathcal{U}_i|}\right\}, \forall c_i \in \mathcal{C}$ includes all associated UEs with the i-th slice.

Then, two global variables are formed in Algorithm 1:

- $\mathbf{C}$ is a vector of size $|\mathcal{C}|$ comprising all slice configurations.
- $\mathbf{S}_i$ is a vector of size $|\mathcal{U}_i| + 1$ including the number of scheduled resources for the i-th slice (i.e., $\mathbf{S}_i\left[0\right]$ as the first element of $\mathbf{S}_i$) and for its associated UEs (i.e., $\mathbf{S}_i\left[j\right]$, $\forall j \in [1, |\mathcal{U}_i|]$).

Finally, $\mathbf{S}_i$, $\forall c_i \in \mathcal{C}$ will be passed to the post-processor as the scheduling results.

Then, three inputs are defined: l denotes the scheduling level, r is the number of available RBs, and set $\mathcal{N}$ contains the unscheduled slices. To start the recursive operation, the following arguments are used: $l \leftarrow slice$, $\mathcal{N} \leftarrow \mathcal{C}$, and $r \leftarrow R$, where R is the maximum number of RBs can be scheduled. At the slice-level from line 4 in Algorithm 1, we first select an unscheduled slice using the $SelectSlice\left(\cdot\right)$ function, allocate RBs to this slice using the $AllocRB\left(\cdot\right)$ function, recursively call SCH to assign RBs to its associated UEs (see line 9), and move to the next unscheduled slice (see line 12). At the user-level from line 15, every associated UE is looped through to schedule all slice-level resources, i.e., $\mathbf{S}_i\left[0\right]$.

2.5.3 Implementation and Testbed Setup

We implement our proposed FlexCtrl architecture on top of a platform comprising components from OAI [114], including the core network, gNB (RAN-Node), and soft-UEs, along with FlexRIC [134], which serves as the NearRT-RIC (RAN Controller) and xApp (App). The abstract control logic is distributed across the xApp, NearRT-RIC, and gNB. Messages exchanged between these components are encapsulated in two customized SMs: SLA and Slice. Moreover, the testbed environment consists of two Ubuntu 20.04.1 machines. The first machine, used for the gNB, UE, NearRT-RIC, and xApp, is equipped with 12 CPUs (3.0 GHz each) and 32 GiB of memory. The second machine, dedicated to the core network, features 18 CPUs (3.7 GHz each) and 31 GiB of memory.

2.6 Performance Evaluation

In this section, we validate the performance of the FlexApp framework and the FlexCtrl architecture through a series of evaluations based on prototypes implemented on state-of-the-art platforms. First, the performance of the FlexApp framework is evaluated in three aspects (see Sections 2.6.1, 2.6.2, and 2.6.3) using a prototype built on the O-RAN-compliant FlexRIC [134]. Specifically:

- **Low-Latency Control Operation and Scalability:** In Section 2.6.1, we measure the control loop latency from the E2-Node to the xApp under different numbers of E2-Nodes, which represents the capability to realize ultra-low latency operations even in a scaled deployment.

- **Superior Performance of E2* Interfcae**: Section 2.6.2 presents a comparison of the proposed E2* interface with two other transport mechanisms, gRPC and RMR/SI95, adopted by the OSC and ONF, respectively (cf. Figure 2.3).

- **Efficient Two-Level Abstract Control Logic**: Section 2.6.3 verifies the efficiency and effectiveness of the xApp control logic after applying the virtualization layer for multi-level abstraction within the FlexApp framework.

Next, we evaluate the FlexCtrl architecture in two aspects (see Sections 2.6.4 and 2.6.5) through a prototype built on the end-to-end 5G stack providing by OAI [114], with controllable radio resource scheduling functions implemented in the OAI gNB:

- **Flexible Control Operation**: Section 2.6.4 provides a quantitative comparison of three CP topologies, analyzing the trade-offs between control operation latency and the additional complexities introduced at the RAN-Node.

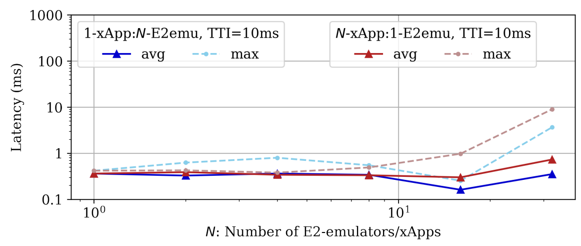
- **Dynamic RAN Slicing**: In Section 2.6.5, we evaluate network and user performance in scenarios where the RAN is dynamically sliced, showcasing the programmability and adaptability enabled by real-time control operations.

2.6.1 Control Loop Latency and Scalability

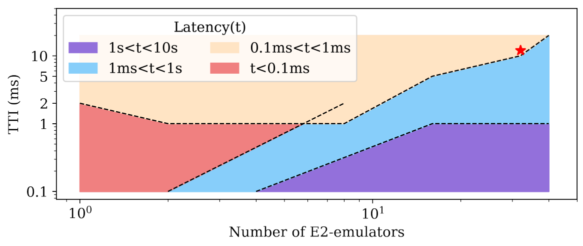
Setup and Workloads. In this evaluation, three Precision Timing Protocol (PTP) synchronized machines are used to deploy the E2-emulator, FlexRIC, and xApp. We cover both containerized and non-containerized environments. The xApp subscribes to the MAC SM and obtains the monitoring data from the E2-emulator(s) via the E2* interface for every Transmission Time Interval (TTI) in a 12-byte message format (including E2AP and E2SM protocols). Two relationships between xApps and E2-emulator are evaluated: 1-to-N and N-to-1.

Low-Latency Control Operation. As shown in Figure 2.9a, both the average and the maximum values of control loop latency are presented for 10 ms TTI in a containerized environment. Therefore, within 1 sec, there are 100 messages to be processed for each connected E2-emulator or xApp. The average control loop latency remains constant when the number of E2-emulators or xApps is low (i.e., 8); however, its values increase when there are either 32 E2-emulators[6] or 32 xApps, but remain below 1 ms for ultra-low latency operations. A similar trend is observed in the maximum control loop latency, where the values are all lower than 10 ms. This guarantees that the FlexApp approach will not produce an internal bottleneck under a 10 ms TTI.

[6]The maximum number of supported E2-emulators is currently limited by FlexRIC; hence, we only evaluate up to 32 E2-emulators.



(a) Control loop latency between xApp(s) and E2-emulator(s).



(b) Feasible regions of different latency restrictions.

Figure 2.9: Latency measurements and scalability evaluations.

Scalability. Based on the results shown in Figure 2.9a, we can conclude that the FlexApp framework can support 32 xApps using its northbound to monitor SM from an E2-Node, and an xApp can simultaneously monitor 32 E2-Nodes connected to the NearRT-RIC. In parallel, we perform a stress test on the FlexApp framework, which shows that up to 350 xApps could be supported per machine, and the average control loop latency is around 6 ms. This limitation is due to the thread mode implemented in our prototype and the way messages are enqueued. We plan to make improvements in the future.

Key Observations. In addition, we measure the control loop latency in a non-containerized environment to plot the viable regions for different latency restrictions, as shown in Figure 2.9b, in terms of various combinations of TTI values and the number of E2-emulators. Take the example case with 1 ms latency restriction, its viable region covers both the light orange and the light red parts in Figure 2.9b, since these two parts can support latency below 1 ms. We can see a tradeoff between how frequently the data can be updated and how many E2-Nodes can be monitored simultaneously, e.g., shorter TTI values must come with fewer monitored E2-Nodes. Also, the red star in the same figure represents the setting used in Figure 2.9a, i.e., 10 ms TTI and 32 E2-Nodes, and this setting is confirmed to be functional for 1 ms latency restriction.

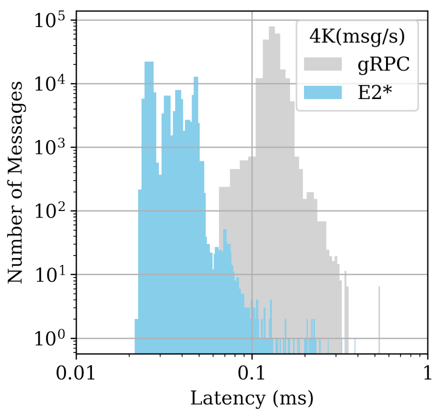
2.6.2 Benchmarking on Transport Mechanism

Setup and Workloads. Different transport mechanisms between NearRT-RIC and xApp are compared in this section. Since our focus is on the transport mechanism, a simple 1-to-1 relationship between xApp and FlexRIC is applied (similar to [122]), but with two different TTI values, i.e., 250 µs and 10 ms. Specifically, an xApp is used to measure the latency by calculating the time difference from when the indication message is forwarded by the NearRT-RIC to when it arrives at xApp callback function.

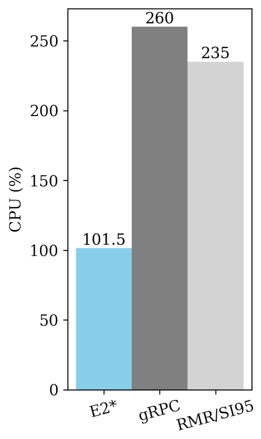
Latency and CPU Usage. In Table 2.3 and Figure 2.3a, both E2* interface and gRPC are compared at different message rates, and the E2* interface outperforms at every percentile value. Furthermore, in Figure 2.3b, we compare the CPU utilization of the xApp to receive the transported messages among the E2* interface, the gRPC, and the RMR/SI95. We can

Table 2.3: Latency comparison in milliseconds.

	Min.	Max.	50%	95%	99%	Total Sent Msg.	Rate (Msg./s)
gRPC	0.07	>1.0	0.15	0.18	0.20	100K	4K
E2*	0.02	0.39	0.11	0.11	0.11	100K	4K
gRPC	0.20	0.54	0.31	0.37	0.39	25K	0.1K
E2*	0.03	0.21	0.06	0.06	0.06	25K	0.1K



(a) Latency comparison: 100K messages sent at 4K rate (msg/s).



(b) CPU usage comparison.

Figure 2.10: Latency and CPU usage comparison of transport mechanisms.

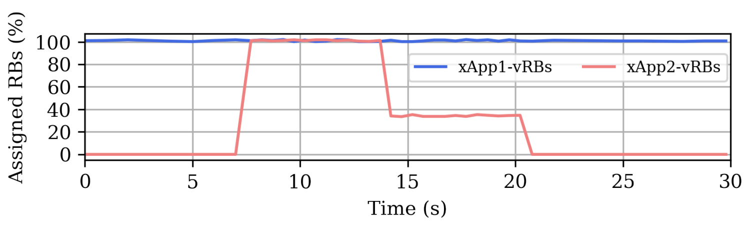
see that the E2* interface consumes considerably less CPU resources (i.e., 101.5% compared with 260% and 235%) than the other two.

Key Observations. In summary, by design, the E2* interface can reach a lower latency than the gRPC because it performs fewer work/abstractions, e.g., a raw SCTP connection in the E2* interface versus an HTTP-based approach in the gRPC. Moreover, the E2* interface consumes less CPU resource at the xApp owing to the simpler FlexApp architecture and fewer messages exchanged between xApp and NearRT-RIC. To provide more details, the xApp using RMR/SI95 must handle the message from the NearRT-RIC for routing management before handling the message, and the message must be encoded/decoded several times between each micro-service.

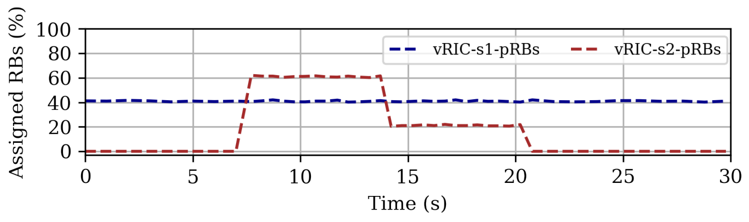
2.6.3 Two-Level Abstract Control Logic

Setup and Workloads. To validate the multi-level abstract control logic approach described in Section 2.4, we build a virtualization layer (see Figure 2.5) in the NearRT-RIC platform and test its ability to interact with two separate slicing control xApps. Based on the respective service requests from these xApps, radio resources from the E2-emulator, i.e., 162 Resource Blocks (RBs) in a 60 MHz bandwidth with numerology 1 can achieve up to 200 Mbps, will be assigned accordingly. Moreover, to compare different perspectives after and before the two-level abstractions (see xApp-II in Figure 2.4), two respective figures are presented in Figure 2.11a and Figure 2.11b.

Abstract Radio Resource. Figure 2.11a (2nd level abstraction) shows the virtual RBs assigned to the xApp by the virtualization layer, whereas Figure 2.11b (1st level abstraction) represents the physical RBs seen from the NearRT-RIC perspective. Initially, only xApp1 is



(a) Percentage of virtual RBs from xApps' perspective.



(b) Percentage of physical RBs from NearRT-RIC's perspective.

Figure 2.11: Assigned RBs percentage in two-level abstract control logic.

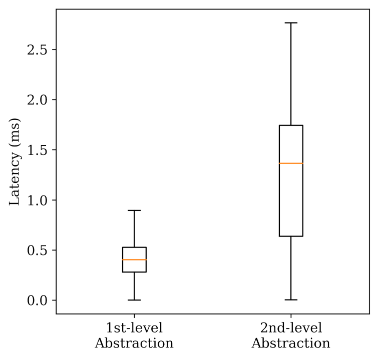


Figure 2.12: Measured latency of control loop.

present, and it needs 80 Mbps for its service. Therefore, only 40% of the physical RBs are requested by the virtualization layer from the E2-emulator (see vRIC-s1-pRBs in Figure 2.11b). But from xApp1's point of view, it assumes it receives 100% of the virtual RBs because of the second-level abstraction (see xApp1-vRBs in Figure 2.11a). At the 7 s, xApp2 is deployed and requests 120 Mbps instead. In this regard, the virtualization layer requests 60% of the physical RBs from the E2-emulator (see vRIC-s2-pRBs in Figure 2.11b), but xApp2 receives 100% of the virtual RBs (see xApp2-vRBs in Figure 2.11a), similar to xApp1. Then, at the 14 s, xApp2 lowers its request to 40 Mbps, and thus the corresponding decreases by a factor of three are made in both Figure 2.11a and Figure 2.11b. Finally, xApp2 releases its request at the 21 s but has no effect on xApp1.

Control Loop Latency. Furthermore, we measure the control loop latency between the E2-emulator and the xApp after considering the virtualization layer as the second-level abstraction. In specific, the measured latency as shown in Figure 2.12 represents the time difference from when the xApp sends a control message until it receives the control acknowledgement. Compared with the measurements conducted in Section 2.6.1, these results are more informative in terms of considering the second-level abstraction as well as the latency statistics for the control loop. We can see that the latency after considering the second-level abstraction is approximately three times higher than that of the first-level abstraction case. However, it still fulfills the requirements for control loops in NearRT-RIC (between 10 ms and 1 sec), implying that extra-level abstractions (e.g., third-level abstraction xApp-III in Figure 2.4) are potentially feasible.

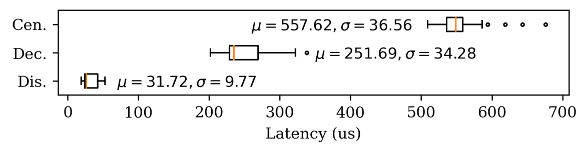


Figure 2.13: Control loop latency comparison.

Table 2.4: Memory consumption comparison.

Control Plane Topology	Cen.	Dec.	Dis.
Control Logic Placement	xApp	NearRT-RIC	E2-Nodes
Baseline Memory	3.41 MB	3.50 MB	1.19 GB
Extra Memory of Control Logic	0.23 MB	0.10 MB	0.26 MB
Number of Control Logic For N E2-Nodes	1	1	N

2.6.4 Trade-Off Between Three Control Plane Topologies

Setup and Workloads. In this section, we quantify the trade-off between three distinct CP topologies - Centralized (Cen.), Decentralized (Dec.), and Distributed (Dis.) - in terms of control loop latency and resource consumption. In each topology, abstract control logic is deployed in different components: the App, RAN controller, and RAN-Node. These correspond to the O-RAN terminology for the xApp (App), NearRT-RIC (RAN controller), and E2-Node (RAN Node), respectively. This control logic processes service requirements defined in the SLA SM (received via service APIs) and translates them into slice configurations in the Slice SM (communicated through RAN APIs).

Latency of Control Operation. First, the control loop latency is measured between the location of abstract control logic and the corresponding Rf in Figure 2.13. We can see that the distributed topology has the lowest mean value and least variation, the reason behind this is that there is no need to leverage the protocol stack, i.e., Stream Control Transmission Protocol (SCTP), between E2-Node and NearRT-RIC as well as between xAPP and NearRT-RIC for communication. Such benefit is essential to realize time-critical business logic into the real-time programmable functions in RAN UP for deterministic behavior. In contrast, the decentralized and centralized topologies take 8x and 17x more latency, due to the extra one-hop and two-hop operations, respectively.

Memory Usage of Control Logic. We then measure the memory usage of xApp, NearRT-RIC, and E2-Nodes in Table 2.4 before and after introducing the abstract control logic. First, the extra memory incurred by realizing the abstract control logic in NearRT-RIC (decentralized topology) is minimal. This is because the NearRT-RIC was originally designed to communicate with xApp and E2-Node via SLA SM and Slice SM, so no extra message handling is required. Also, when we compute the ratio of the extra memory usage to the baseline memory usage, the smallest result occurs in the distributed topology. This is because an E2-Node already requires a larger baseline memory to accommodate all RFs. But when a single business logic would like to control N E2-Nodes (N>1), this extra memory usage for abstract control logic needs to be deployed in every E2-Node due to the distributed nature.

Key Observations. To sum up, there are pros and cons to deploying control logic in different locations. First, the centralized topology enables the xApp to directly define its requirements for all RAN-Nodes, and it can naturally control multiple E2-Nodes without additional resources (cf. Table 2.4), but at the cost of a higher control loop latency (cf. Figure 2.13). In contrast, the decentralized topology requires minimal resources and can control multiple underlying E2-Nodes without further resources (cf. Table 2.4). Also, the control loop latency is reduced compared to the centralized one (cf. Figure 2.13). Finally, the distributed topology shows minimal control loop latency with the smallest variance to enable deterministic business logic; however, control logic abstraction needs to be performed at each E2-Node.

2.6.5 Use Case: Dynamic RAN Slicing

Setup and Workloads. In this section, we aim to show the capability of FlexCtrl for the use case - dynamic RAN slicing - when applying different algorithms and changing parameters on-the-fly. Specifically, a single E2-Node is used to serve UE1 and UE2, and its maximum number of schedulable RBs is R=106. Moreover, due to radio condition, the maximum cell capacity is about 130Mbps, and we send fixed 120Mbps downlink User Datagram Protocol (UDP) traffic to each UE. It is worth emphasizing that this scenario is created to observe the impacts on dynamic RAN slicing even when RAN UPs is fully loaded. Moreover, six scenarios are described in Table 2.5 to serve both UE1 and UE2, with the number of slices varying between zero (Scenario 1), one (Scenarios 2 and 5) and two (Scenarios 3, 4, and 6). Also, both NVS [100] and EDF [86] algorithms apply different parameters in these scenarios: NVS uses the cap parameter to identify the slice radio resource share in percentage, while EDF uses the d parameter to indicate the maximum delay and the n parameter to denote the number of RBs provided during this period[7].

As follows, we go through the results of each scenario in terms of RB utilization percentage and MAC scheduler processing time at E2-Node (Figure 2.14a), the perceived throughput at UE (Figure 2.14b), and both one-way delay and packet loss percentage (Figure 2.14c):

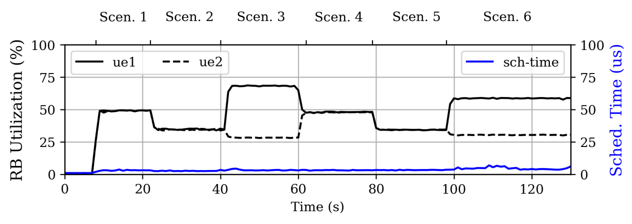
Scenario 1. At 5 sec, both UE1 and UE2 consume 50% of RBs because no slicing is employed and the PF algorithm is used at the user-level. Since the sum of UP traffic (240 Mbps) is much higher than the cell capacity (130 Mbps), the one-way delay and loss rate are high, but seem fair to both UEs.

Scenario 2. At 22 sec, we use the NVS algorithm and set 70% of the radio resources to slice 1, so that each UE takes 35% of the radio resources. Moreover, both one-way delay and loss rate increase accordingly (around 1.3x) due to traffic overload of a finite queue size (i.e., first-in, first-out blocking queue).

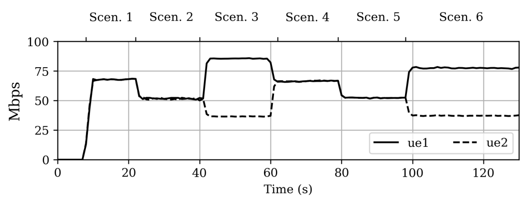
[7]The EDF algorithm creates a priority list of network slices based on their current deadline, but the number of RBs will not exceed n.

Table 2.5: Description of six scenarios (Here s1 and s2 represent slice 1 and 2, respectively).

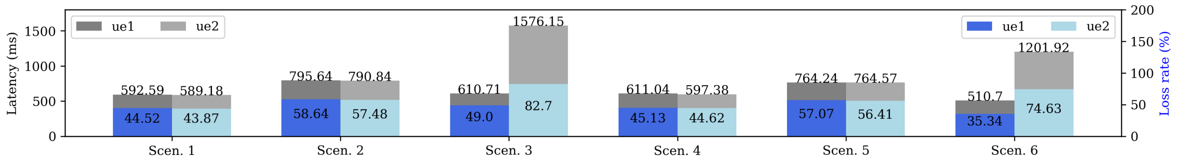
Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
No Slicing	s1 (NVS, *cap*=70): UE1, UE2	s1 (NVS, *cap*=70): UE1 s2 (NVS, *cap*=30): UE2	s1 (NVS, *cap*=50): UE1 s2 (NVS, *cap*=50): UE2	s1 (EDF, *d*=2, *n*=150): UE1, UE2	s1 (EDF, *d*=2, *n*=150): UE1 s2 (EDF, *d*=20, *n*=620): UE2



(a) RB usage and MAC scheduler processing time.



(b) Downlink throughput of two UEs.



(c) One-way delay and packet loss percentage.

Figure 2.14: Network and user performance in a fully-loaded scenarios with RAN controlled by an xApp in centralized control plane topology.

Scenario 3. At 40 sec, the second slice is set up with 30% of the radio resource for UE2; thus, UE1 will occupy all 70% of the radio resource for slice 1. As for the one-way delay in Figure 2.14c, it is inversely proportional to *cap* values, but UE1 has a larger latency than in Scenario 1. This is because the NVS algorithm selects one slice at a time, so higher throughput (cf. Figure 2.14b) does not imply lower delay (cf. Figure 2.14c).

Scenario 4. At 62 sec, both slices use *cap*=50, so compared to Scenario 1, both UEs use a similar number of RBs and have similar throughput and loss rates. Due to the above characteristic of the NVS algorithm, the one-way delay is slightly higher than Scenario 1.

Scenario 5. At 79 sec, the EDF algorithm is applied and the second slice is removed. Note that the EDF algorithm will preserve n=150 RBs within the maximum delay of d=2 slots. So about 70% of RBs will be allocated to Slice 1, similar to Scenario 2; however, this scenario has a lower one-way delay because the preserved resources are fixed periodically, which is not the case for the NVS algorithm.

Scenario 6. At 98 sec, the second slice is set up, and it requests about 30% of RBs in a relaxed deadline d=20. In Figure 2.14a, UE1 takes about 60% of RBs, while UE2 occupies 30% of RBs. It is worth noting that about 10% of RBs are unused, since we do not over-provide the values of n, which is smaller than the maximum number of schedulable RBs per slot. We also see that UE1 now has a lower latency and loss rate than Scenario 1, because it has a smaller deadline and thus takes precedence over UE2 most of the time, while both

UEs are only handled by the PF algorithm in Scenario 1.

In short, the FlexCtrl framework enables dynamic RAN slicing no matter what algorithms or parameters are changed on-the-fly. Also, the redesigned radio resource scheduler has a very small footprint (cf. Figure 2.14a), even when RAN is fully loaded.

2.7 Discussion and Future Work

The performance evaluation of both the FlexApp framework and FlexCtrl architecture demonstrates their combined potential to achieve the vision of a “Flexible Control Plane” for modern and future SD-RAN systems.

The FlexApp framework demonstrates its suitability for seamless integration with the NearRT-RIC platform, outperforming other xApp frameworks by delivering lower latency and reduced CPU usage while introducing novel functionalities like two-level abstraction. These features simplify xApp development by abstracting raw data into business logic through a virtualization layer, enabling ultra-low latency operations supported by the innovative E2* interface. Informative results, such as the feasible region (Figure 2.9b) and control loop latency (Figure 2.12), provide valuable insights for optimizing xApp deployments across different RAN configurations.

Complementing FlexApp, the FlexCtrl architecture enhances flexibility by enabling dynamic RAN slicing with exceptional adaptability. It supports real-time changes to slicing parameters, algorithms, and UE-slice associations without performance degradation or system interruptions. Comparative analysis of three CP topologies reveals distinct trade-offs. The centralized topology, widely adopted in current O-RAN architectures, offers full control over business logic implementation but at the expense of higher latency and tighter App-platform dependencies. The decentralized topology achieves lower control loop latency and reduced resource usage while managing multiple E2-Nodes efficiently. The distributed topology, with its minimal latency ($<$50 µs as shown in Figure 2.9a) and deterministic real-time control, is ideal for time-sensitive applications in smaller E2-Node clusters. Together, FlexApp and FlexCtrl work synergistically to deliver a flexible CP that meets the demands of diverse use cases.

Looking ahead, future work will focus on scaling the northbound interface of control applications to support more complex, multi-tier, and multi-RAT RAN topologies while leveraging the virtualization layer for enhanced abstraction. This includes expanding the flexibility of control applications and exploring new functionalities, such as interference management, to improve RAN coordination. Additionally, porting existing control applications, such as traffic steering, will enhance compatibility and interoperability across RAN controller platforms.

2.8 Conclusions

In this chapter, we introduced the FlexApp framework and the FlexCtrl architecture as integral components for achieving a flexible CP in future O-RAN and SD-RAN systems, tailored to meet emerging use cases.

The FlexApp framework is designed to support latency-sensitive RFs while enabling multi-level abstractions and utilizing a future standardized E2* interface for seamless communication between xApps and the NearRT-RIC. Its compatibility with various NearRT-RIC platforms ensures broad applicability. Performance evaluations of the FlexApp prototype highlight its key advantages, including ultra-low latency operation, high scalability, minimal overhead, and advanced abstraction capabilities.

The FlexCtrl architecture extends control flexibility and enhances the real-time programmability of the RAN functions, surpassing the current limitations of the O-RAN architecture. By supporting three distinct CP topologies, FlexCtrl enables the customization of control logic placement based on scenario-specific latency requirements. The redesigned RAN functions further enable dynamic RAN slicing through abstracted control logic. Prototype evaluation confirms that FlexCtrl supports all three topologies, achieves real-time control capability (under 50 µs in the distributed topology), and enables dynamic RAN slicing without performance degradation.

These contributions mark significant progress toward a flexible CP, setting a strong foundation for innovations in 5G-Advanced networks and providing the essential building blocks for future 6G systems.

Chapter 3

Integrated and Programmable User Plane

3.1 Introduction

The Fifth Generation (5G) system is designed with evolving principles, regularly upgraded with each generation to support not only the Third-Generation Partnership Project (3GPP) services but also IP-based services, such as web browsing and other forms of Internet access [18]. However, with the rapid growth of the application ecosystem and the rise of over-the-top services, application traffic has become the primary source of network usage for end-users [80]. Unlike the relatively stable evolution of mobile network infrastructure, application traffic characteristics often change faster due to frequent software updates and new releases, outpacing the typical ten-year mobile network upgrade cycle.

In this context, we envision that next-generation mobile networks will embrace an application-centric design, characterized by a flattened and routing/switching-based architecture, to enable a converged network [92]. This new focus will enhance interconnections between points of presence and provide real-time information for application decision-making. From an application perspective, it can interface with connected access network technologies (e.g., WiFi) and network infrastructure elements (e.g., router), allowing applications to optimize traffic delivery path(s) and improve the efficiency.

As for today, achieving this goal in 5G faces three challenges: First, separating Radio Access Network (RAN) functions, either by splitting the Distributed Unit (DU) and Centralized Unit (CU) or dividing the User Plane (UP) from the Control Plane (CP), can cause inconsistencies between traffic control policies and radio resource scheduling. Second, RAN operates on aggregated service data flows called Data Radio Bearers (DRBs), which limits its ability to provide precise link-layer information of each flow to applications. Third, tunneling and encapsulation between CU-UP and User Plane Function (UPF) add processing overhead, introduce latency, and hinder direct IP routing for local User Equipment (UE) communications.

Beyond the challenges in 5G UP, enabling RAN programmability follows different approaches. Architecturally, the O-RAN framework [127] enables dynamic RAN control through xApps on the Near-Real Time RAN Intelligent Controller (NearRT-RIC), handling tasks like radio resource allocation [64]. From an optimization perspective, cross-layer coordination improves traffic management, addressing issues like buffer bloat [90] and prioritizing short flows [98]. However, achieving a fully programmable RAN that adapts to real-time network conditions requires integrating both architectural control and optimization techniques.

Contributions. To this end, this chapter aims to foster a discussion on integrating UP across RAN and UPF into a unified entity. As shown in Figure 3.1, we introduce the concept of Integrated User Plane (IUP), envisioned as the out-of-the-box deployment for next-generation mobile networks. IUP not only simplifies mobile network architecture by reducing latency and protocol overhead, e.g., GPRS Tunneling Protocol User Plane (GTP-U), but also enables programmability from IP flow traffic control to radio resource allocation. In Section 3.3, we outline the implications and potentials of IUP for next-generation mobile networks. Furthermore, in Section 3.4, the detailed design of IUP is provided and compared with the current 5G system. We also elaborate on the programmability provided by IUP to other entities, e.g., NearRT-RIC and Core Network (CN). Finally, we explore some key use cases (Section 3.5) of IUP deployment and evaluate its performance through a proof-of-concept implementation (Section 3.7), showing how IUP can reduce network latency, converge seamlessly with non-3GPP network, and provide UP programmability.

3.2 Related Work

Building on the SDN paradigm, SD-RAN unlocks the programmability of RAN functions, encompassing both the CP and User Plane (UP)[1]. Several SD-RAN platforms [83, 96, 134] have been developed to enhance the real-time programmability and customizability of RAN functions.

FlexRIC [134] supports both standardized and customized Service Models (SMs) using technologies such as shared libraries, enabling a plugin-based SM architecture. This approach allows RAN and application vendors to select and load specific SMs at runtime, enhancing the scalability of RAN functionalities while reducing the overhead of unused SMs. Similarly, HexRAN [96] also introduces a plugin-based SM architecture and develops a Programmable Mediation Layer to provide Application Programming Interfaces (APIs) that simplify O-RAN related operations within 3GPP RANs. Meanwhile, JANUS [83] provides a sandboxed execution environment based on eBPF [144], allowing customized codelets to be deployed at runtime to enhance RAN telemetry, collect arbitrary statistics, and perform real-time inference to assist in control decisions, while offering a secure mechanism for user-defined telemetry collection.

However, existing solutions primarily focus on the programmability of functionalities within the RAN node, while neglecting coordination with Core Network (CN) components, such as the User Plane Function (UPF). This lack of integration limits the ability of mobile system to optimize end-to-end network performance and effectively manage UP data flows. Moreover, some efforts [60, 89, 110] have been made to optimize and enable programmbility of RAN backhaul data flows at the UPF level by leveraging technologies such as eBPF or P4 [61]. However, these approaches lack coordination with real-time radio link information, such as RLC buffer status and MAC radio resource scheduling, leading to unsynchronized optimizations. Although some work [98] addresses this issue by analyzing packet information within the RAN, its approach is limited to prioritizing small packets and lacks the programmability needed to dynamically control RAN functionalities. To address these challenges, programmability must be enabled for both the UPF and RAN, fostering a more adaptive network capable of effectively managing UP data flows while adapting to fluctuating radio conditions and dynamic application behavior. In contrast to existing solutions, the proposed IUP architecture facilitates tight coordination between IP traffic flow control and radio resource allocation by integrating UPF functionalities directly into the RAN node.

In parallel, recent work from the Internet Engineering Task Force (IETF) has proposed mapping 5QI values to Differentiated Services Code Point (DSCP) markings to enforce 5G network slice QoS within the DiffServ architecture [74, 87]. Additional studies have focused on improving the granularity of 5G QoS frameworks [139, 149]. Instead of relying on

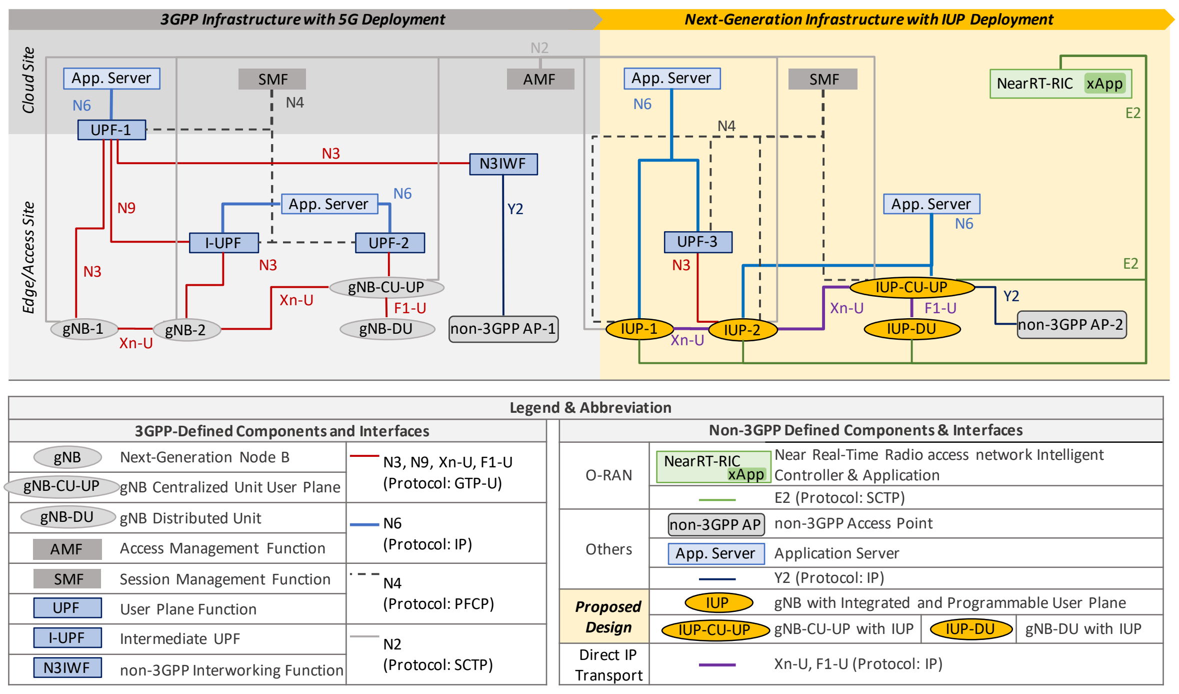
[1]In telecom terminology, the UP corresponds to the data plane.

static 5QI values to manage diverse application flows, the proposed IUP architecture handles native IP packets through its Integrated Data Flow Control (IDFC) sublayer, enabling consistent and adaptable QoS treatment. This design aligns with the PDU Set concept introduced in 3GPP Release 18 [18], which offers finer granularity for managing IP flows. IUP further supports this by leveraging its traffic control pipeline within the IDFC to segregate individual PDU sets into separate queues.

In terms of user plane tunneling, 5G networks adopt various strategies to handle IP fragmentation, including GRE, VXLAN, and Geneve [141]. As highlighted in [141], GTP-U over the N3 interface introduces considerable overhead due to increased MTU requirements and additional header fields. This overhead can be mitigated by transmitting IP frames over Ethernet using a switched tunnels controlled by SDN [78]. IUP eliminates N3-related processing and tunneling, thereby simplifying the infrastructure and enabling native IP support within the RAN.

3.3 Why IUP?

Based on the above vision provided by IUP, below we identify three key benefits and potential improvements.



Legend & Abbreviation

3GPP-Defined Components and Interfaces			
gNB	Next-Generation Node B	🟥	N3, N9, Xn-U, F1-U (Protocol: GTP-U)
gNB-CU-UP	gNB Centralized Unit User Plane		
gNB-DU	gNB Distributed Unit	🟦	N6 (Protocol: IP)
AMF	Access Management Function		
SMF	Session Management Function	- - -	N4 (Protocol: PFCP)
UPF	User Plane Function		
I-UPF	Intermediate UPF	⬜	N2 (Protocol: SCTP)
N3IWF	non-3GPP Interworking Function		

Non-3GPP Defined Components & Interfaces				
O-RAN	NearRT-RIC xApp	Near Real-Time Radio access network Intelligent Controller & Application		
	🟩	E2 (Protocol: SCTP)		
Others	non-3GPP AP	non-3GPP Access Point		
	App. Server	Application Server		
	🟦	Y2 (Protocol: IP)		
Proposed Design	IUP	gNB with Integrated and Programmable User Plane		
	IUP-CU-UP	gNB-CU-UP with IUP	IUP-DU	gNB-DU with IUP
Direct IP Transport	🟪	Xn-U, F1-U (Protocol: IP)		

Figure 3.1: From 5G disaggregated user plane deployment to IUP in next-generation mobile networks.

3.3.1 Reduced Protocol Overhead, Processing, and Latency

5G RAN UP includes not only sublayers for air-interface communication, e.g., Service Data Adaptation Protocol (SDAP), Packet Data Convergence Protocol (PDCP), Radio Link Control (RLC), Medium Access Control (MAC) and Physical (PHY), but also functions like GTP-U processing for tunneling to UPF over N3 interface. In 5G, centralized UPF and distributed RAN are commonly used, resulting in a "long N3" (see gNB-1 and UPF-1 in Figure 3.1). However, advances in RAN cloudification and Mobile Edge Computing (MEC) [2], enables a distributed UPF that shortens N3 link by deploying UPF or Intermediate UPF (I-UPF) [18] along with application servers at the edge/access sites (see gNB-2 and I-UPF, as well as gNB-DU, gNB-CU-UP and UPF-2 in Figure 3.1).

Distributing the UPF closer to the edge is a key approach to enabling low-latency applications and achieving more efficient UP processing in the mobile network backhaul [125]. However, this approach does not eliminate the fundamental network delay caused by physical separation and instead introduces additional network hops and overhead. A detailed comparison of these technologies is provided in Table 3.1.

Table 3.1: State-of-the-art comparison for mobile network deployment technology.

Deployment Technologies	**Solution Placement**	**Min. Network Hops in UP Path**	**Extra Overhead of Inner-IP App. Packet**
Legacy UPF	-	2	GTP-U, UDP, and Outer-IP Headers
Intermediate UPF (I-UPF) [18]	Backhaul	3	GTP-U, UDP, and Outer-IP Headers
Mobile Edge Computing (MEC) [2]	Backhaul	2	GTP-U, UDP, and Outer-IP Headers
Network In a Box (NIB) [128]	Backhaul	2	GTP-U, UDP, and Outer-IP Headers
RAN Disaggregation (or Open RAN) [147]	Midhaul & Fronthual	3	GTP-U, UDP, and Outer-IP Headers
Integrated Access Backhaul (IAB) [8]	Midhaul	3	BAP Payload and GTP-U, UDP, and Outer-IP Headers
IUP	**Midhaul & Backhaul**	**1**	**None**

One notable method, I-UPF [18], introduces an intermediary hop between the centralized UPF and the RAN. Similarly, MEC [2] supports edge-localized deployment of UPF and application servers, though this can increase network overhead and complexity. Another innovative solution, Network In a Box (NIB) [128], consolidates entire core network functions with RAN within the same physical unit at the access site, further pushing the boundary of edge computing. Despite these advancements, the fundamental delay between UPF and RAN persists, primarily due to the physical separation inherent in their deployment.

From another perspective, such delay negatively affects several emerging technologies, including Open RAN [147] and Integrated Access Backhaul (IAB) [8]. Open RAN, also referred to as RAN disaggregation, splits the gNB into DU and CU, increasing the number of hops in the UP. Similarly, IAB introduces another protocol, the Backhaul Adaptation Protocol (BAP) for communication between IAB nodes, adding an additional layer of protocol-induced delay to the UP. In both cases, communication between network entities relies on GTP-U tunneling, such as between CU and DU over the F1-U interface and between IAB nodes. This dependency on GTP-U introduces additional processing overhead, adding latency and reducing efficiency in the UP.

Building on this, the proposed IUP removes the need for the N3 interface and the corresponding GTP-U processing (e.g., en/de-capsulation), as shown by IUP-1, IUP-DU, and IUP-

CU-UP in Figure 3.1. Additionally, it avoids IP fragmentation caused by different Maximum Transmission Unit (MTU) sizes between GTP Protocol Data Unit (PDU) and IP packet [78, 141]. IUP also supports the deployment with an extra hop to UPF and application server, serving as an intermediate stage of deployment to prevent an overall network renewal (see IUP-2 in Figure 3.1). In terms of latency reduction, solutions like network slicing have already shown their capability to meet the hierarchical Quality of Service (QoS) requirements specified in Service Level Agreements (SLAs) [5]. Note that IUP operates independently of network slicing, it reduces latency from two sources. First, by removing backhaul transmission delays to centralized/distributed UPF, IUP better manages the Packet Delay Budget (PDB) [18], not only from a RAN perspective but also across backhaul network. Second, it bypasses N3-related GTP-U processing.

3.3.2 Programmable User Plane & Simplified Control Plane

Existing solutions for UP programmability typically focus on two approaches: (1) introducing programmability to the packet processing pipeline in the UPF [20], such as P4 on programmable hardware platform [110, 60]; and (2) leveraging O-RAN framework to develop control applications (e.g., xApps on NearRT-RIC [127]) for dynamically controlling RAN functions [64]. Therefore, for programmable UP in mobile networks, these two components must coordinate to avoid conflict control decisions. Thanks to the integration of UPF into RAN, IUP addresses this challenge by enabling programmable IP packet processing while simultaneously controlling lower-layer processing of the mapped DRBs.

Moreover, IUP is designed to be compatible with existing CP functions of the CN, as shown in the right part of Figure 3.1. It retains Packet Forwarding Control Protocol (PFCP) functionalities, allowing it to interact with Session Management Function (SMF) via the N4 interface and reuse PDU session management procedures. Also, the N2 interface toward Access and Mobility Management Function (AMF) is preserved for connection management. As IUP becomes an anchoring point for data delivery, it can avoid extra control messages and data forwarding between UPFs and RANs in some scenarios, e.g., inter-gNB handover and home-routed roaming.

3.3.3 Seamlessly Converged Network

Network convergence with non-3GPP networks can be achieved in different ways. One approach is to design a compatibility layer or network function, such as Non-3GPP Interworking Function (N3IWF) [1], to connect 5G and non-3GPP networks (see non-3GPP AP-1 in Figure 3.1). Alternatively, incorporating IP into RAN to reach Internet-wide routing and addressing is another option that can provide interoperability regardless of underlying link technologies. Therefore, IUP adopts this approach by natively supporting IP within the RAN and functioning as a Layer 3 device, inherently facilitating seamless connectivity with non-3GPP networks. This design enables real-time applications to be aware of latency across multiple paths to destination, facilitating different traffic flows (e.g., extended reality objects) using individual routes. As illustrated in Figure 3.1, IUP and non-3GPP AP-2 can seamlessly converge without requiring an additional gateway in between.

Overview of IUP Benefits. First, the proposed IUP reduces N3-related overhead and processing while also simplifying the end-to-end data delivery path in the mobile network (cf. Section 3.7.1). Additionally, IUP enhances UP programmability by extending control from the IP layer (and above) to the radio link layer, providing a unified framework for managing both packet flows and radio resources (cf. Section 3.7.3). Finally, IUP enables universal connectivity by acting as a Layer 3 device, seamlessly integrating diverse access technologies via the IP protocol (cf. Section 3.7.2).

3.4 IUP Architecture

In this section, as shown in the lower half of Figure 3.2, the architecture of IUP is presented, corresponding to IUP-1 in Figure 3.1. For comparison, the legacy deployment is depicted in the upper half of Figure 3.2, corresponding to gNB-1 in Figure 3.1. The proposed architecture enables native IP processing within RAN through the Integrated Data Flow Control (IDFC) sublayer, allowing the execution of UPF functionalities directly on top of RAN protocol stacks. Figure 3.2 highlights how user data are handled in the UP functions and outlines its interaction with two key network entities: SMF, as defined in 3GPP CN, and NearRT-RIC, as part of the O-RAN framework, via their respective CP interfaces.

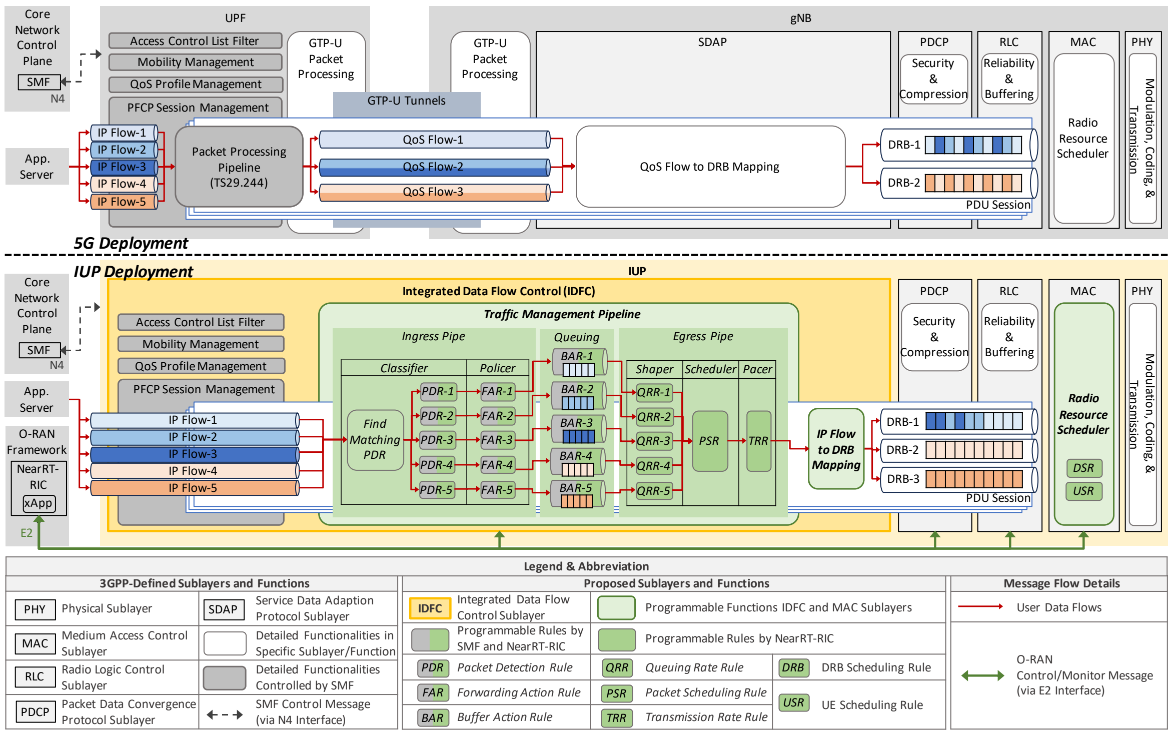

Core Network Control Plane
SMF
N4
UPF
Access Control List Filter
Mobility Management
QoS Profile Management
PFCP Session Management
GTP-U Packet Processing
GTP-U Tunnels
gNB
SDAP
PDCP
RLC
MAC
PHY
Security & Compression
Reliability & Buffering
Radio Resource Scheduler
Modulation, Coding, & Transmission
App. Server
IP Flow-1
IP Flow-2
IP Flow-3
IP Flow-4
IP Flow-5
Packet Processing Pipeline (TS29.244)
QoS Flow-1
QoS Flow-2
QoS Flow-3
QoS Flow to DRB Mapping
DRB-1
DRB-2
DRB-3
PDU Session
5G Deployment
IUP Deployment
IUP
O-RAN Framework
NearRT-RIC
xApp
E2
Integrated Data Flow Control (IDFC)
Traffic Management Pipeline
Ingress Pipe
Queuing
Egress Pipe
Classifier
Policer
Shaper
Scheduler
Pacer
Find Matching PDR
PDR-1
PDR-2
PDR-3
PDR-4
PDR-5
FAR-1
FAR-2
FAR-3
FAR-4
FAR-5
BAR-1
BAR-2
BAR-3
BAR-4
BAR-5
QRR-1
QRR-2
QRR-3
QRR-4
QRR-5
PSR
TRR
IP Flow to DRB Mapping
DSR
USR
Legend & Abbreviation
3GPP-Defined Sublayers and Functions
PHY Physical Sublayer
MAC Medium Access Control Sublayer
RLC Radio Logic Control Sublayer
PDCP Packet Data Convergence Protocol Sublayer
SDAP Service Data Adaption Protocol Sublayer
Detailed Functionalities in Specific Sublayer/Function
Detailed Functionalities Controlled by SMF
SMF Control Message (via N4 Interface)
Proposed Sublayers and Functions
IDFC Integrated Data Flow Control Sublayer
Programmable Rules by SMF and NearRT-RIC
Programmable Functions IDFC and MAC Sublayers
Programmable Rules by NearRT-RIC
PDR Packet Detection Rule
FAR Forwarding Action Rule
BAR Buffer Action Rule
QRR Queuing Rate Rule
PSR Packet Scheduling Rule
TRR Transmission Rate Rule
DRB DRB Scheduling Rule
USR UE Scheduling Rule
Message Flow Details
User Data Flows
O-RAN Control/Monitor Message (via E2 Interface)

Figure 3.2: Architecture for 5G and IUP deployments: user data flows, programmable functions and rules, and control plane interfaces.

3.4.1 Bringing IP Flow Processing into RAN

The proposed architecture introduces an IDFC sublayer that not only consolidates UPF functionalities into RAN but also extends the 3GPP-defined packet processing pipeline [20] to allow for programmability through the whole UP, i.e., from IP traffic control to radio resource allocation. IDFC operates at the same level as SDAP in the RAN protocol stack[2]. It allows direct execution of UPF functionalities within IUP by eliminating GTP-U tunneling and prioritizing IP flows as first-class citizens — unlike legacy gNBs, which primarily focus on QoS flows. In particular, Figure 3.2 depicts how user data flows[3] are delivered from the application server, via several RAN sublayers, and finally to end-users.

5G Deployment. As seen in the upper half of Figure 3.2, user data is transmitted over the PDU session, and the context of PDU session is addressed by PFCP session management to determine how to process packets within the 3GPP-defined packet processing pipeline [20]. In details, SMF can control these pipelines by defining rules, e.g., QoS Enforcement Rules (QERs) [20], to decide how UPF translates IP flows into QoS flows and assigns QoS attributes (e.g., 5QI [18]). Subsequently, QoS flows are transmitted from UPF to gNB via GTP-U tunnels, and gNB assigns them to DRBs at the SDAP sublayer. Finally, the remaining processing is performed by gNB, such as PDCP packet compression, RLC buffering, and MAC resource allocation.

Note that while SMF can control the packet processing pipeline in the UPF by relying on the rules and charging policies provided by Policy Control Function (PCF), it does not have visibility of lower-layer information (e.g., RLC buffer status). As a result, SMF cannot detect issues like bufferbloat, where increased queuing delays occur at gNB, despite efficient packet handling in the mobile network backhaul. On the other hand, the MAC radio resource scheduler lacks insight into each application flow because multiple IP flows are aggregated into a QoS flow, and multiple QoS flows are mapped into a DRB (e.g., IP flow-2 and flow-3 are aggregated into QoS flow-2, and QoS flow-1 and flow-2 are mapped into DRB-1, as shown in upper half of Figure 3.2). Therefore, the UP is not properly integrated for unified programmability.

IUP Deployment. In contrast, as shown in the lower half of Figure 3.2, IUP integrates UPF into RAN as the IDFC sublayer while maintaining certain UPF functionalities, such as PDU session establishment, modification, and release via PFCP session management. In this sense, similar to 5G deployment, user data is still sent over PDU sessions. Moreover, the IDFC sublayer can directly handle IP flows, provide granular traffic control in the traffic management pipeline, and map them into DRBs. This pipeline is inspired by Linux traffic control [90], and it has three main stages with several programmable rules: (1) Ingress pipe (classifier and policer), (2) Queuing, and (3) Egress pipe (shaper, scheduler, and pacer), which will be discussed in the next paragraph. In short, IUP eliminates the need for extra GTP-U processing and intermediate QoS flow translation.

[2]The IDFC sublayer replaces the current SDAP sublayer and is not backward compatible; however, such compatibility can be maintained by deploying extra GTP-U and SDAP processing in IUP (cf. Section 3.5.5).

[3]For simplicity, only the downlink user data flows are depicted, since the uplink one are not affected.

3.4.2 Programmable Functions and Rules in User Plane

IUP provides programmability that enables holistic control and monitoring of user data within a single network entity, spanning from the IP level down to the radio resource level. This programmability is achieved via O-RAN framework, in a manner similar to the programmability offered by P4 [110]. While P4 focuses mainly on packet processing within UPF, IUP extends programmability to interact with radio link functions, such as RLC buffer status and radio resource allocation.

IP Level. The traffic management pipeline in IDFC consists of three main stages and can be programmed with six different rules. First, in the ingress pipe, incoming IP flows are identified by classifiers based on Packet Detection Rules (PDRs) and then forwarded or dropped by policers based on Forwarding Action Rules (FARs). Afterwards, packets will be buffered in queues, and each queue will be managed according to the Buffer Operating Rules (BAR). Finally, in the egress pipe, the rate of each queue will be controlled by the shaper using Queuing Rate Rules (QRR), the scheduler will use Packet Scheduling Rules (PSR) to determine which flows can be transmitted, and the Transmission Rate Rule (TRR) is used by the pacer to decide how to pace packets out. In particular, these six rules are elaborated as below:

- **PDRs** identify IP flows using given information, such as five tuples, deep packet inspection, or machine-learning-based models. Within multiple PDRs, the classifier can analyze the packet to find a matching PDR (and destination queue) and then send it to the policer if a matching PDR is found. Otherwise, the packet will be discarded if no default PDR is provided.
- **FARs** are linked to a queue and determine the actions to be taken by the policer before entering the queue, e.g., forwarding or dropping. If no FAR is provided, the packet will bypass this stage and go directly to the destination queue.
- **BARs** provide queue management rules for each queue, e.g., First-In-First-Out (FIFO) and Controlled Delay (CoDel) [113]. And there is a default queue for processing packets that match default PDR.
- **QRRs** are used by the shaper to limit the maximum egress rate of corresponding queue.
- **PSR** provides the approach, such as round-robin or priority-based, to schedule packets across multiple queues by the scheduler.
- **TRR** is used by the pacer to control inter-packet time to avoid unnecessary queuing or even packet dropping in lower layer (e.g., RLC buffer).

Some rules — such as PDRs, FARs, and BARs — are the same as those used in the 3GPP-defined packet processing pipeline specified by SMF. In IUP, these rules are extended for use in the traffic management pipeline controlled via O-RAN framework. Despite this extension, SMF retains control over these rules to perform the necessary CP actions determined by CN (e.g., charging policy). Moreover, IUP offers extensibility in the traffic management pipeline, which can be implemented using advanced technologies such as DPDK or eBPF/XDP to enable fast in-kernel packet processing.

Radio Bearer and Radio Resource Levels. Subsequently, IP flows are assigned to the corresponding DRB (programmable in the O-RAN framework) for the transmission of the radio link. To ensure consistent management from IP flows to DRBs, this cross-layer control logic can be integrated into upper-layer control applications (e.g., xApps) that monitor states like application types, buffer status, and radio resource usage to direct flows to appropriate DRBs. For example, in Figure 3.2, IP flow-1, IP flow-2, and flow-3 are mapped to the same DRB because they have similar traffic management rules; however, IP flow-4 and IP flow-5 are mapped separately due to their distinct rules. Finally, the MAC radio resource scheduler, which allocates radio resources to DRBs and UEs, can also be programmed using DRB Scheduling Rule (DSR) and UE Scheduling Rule (USR), respectively.

To summarize, the whole UP, from IP flows to radio resource scheduling, can be programmed using both SMF and/or NearRT-RIC to fulfill a variety of application needs while optimizing resource utilization and flexibly managing traffic.

3.4.3 Control Plane Interfaces

An important aspect of IUP is its approach to handling control messages, which operates over two CP interfaces: (1) N4 interface between SMF and IUP for 3GPP control messages to manage PDU sessions, and (2) E2 interface between NearRT-RIC and IUP for O-RAN control messages to manage the aforementioned programmable rules.

3GPP N4 Interface. As for the first interface, after UE association and authentication, SMF selects an IUP to allocate IP addresses during the establishment of the PDU session. Once an IUP is selected, SMF communicates with it via PFCP functionalities for tasks such as PDU session establishment, modification, and release, as well as IP anchoring, while also enforcing policies set by PCF. From the perspective of SMF, IUP acts as a common UPF. Additionally, SMF can choose different IUP instances based on service requirements and send control messages to the source or target IUP to perform roaming and mobility management operations. In this case, the control procedures for IUP are simplified by avoiding extra messaging between UPF and RAN (see details in Section 3.5).

O-RAN E2 Interface. For the second interface, xApps on NearRT-RIC control both programmable functions and rules within IUP, as mentioned in the previous subsection. xApps act as intermediary, managing the complex control logic between the diverse application traffic flows and fluctuating radio conditions, making real-time control decisions to meet the requirements of each application. By integrating UPF into RAN, IUP enables decision-making across multiple layers simultaneously, i.e., from traffic management rules of each IP flow to radio resource scheduling of each DRB/UE, and can be simply combined with other schemes, e.g., network slicing. In consequence, decision conflicts can be largely avoided, allowing for efficient processing of IP flows and improved overall network performance.

3.5 Use Cases

This section examines the benefits of IUP by detailing its procedures and architecture, illustrating how it reduces overhead and complexity in next-generation mobile networks through four key use cases: (1) Handover, (2) Roaming, (3) RAN disaggregation and non-3GPP network interworking, and (4) Compatibility with existing UPF.

3.5.1 Handover Procedure

The handover process [11] in mobile networks enables the transfer of ongoing data sessions from one cell to another. However, as the corresponding UPF may be reallocated, both handover processes are analyzed as below, i.e., with or without UPF reallocation[4] as depicted in Table 3.2. In details, the handover process consists of three stages: (1) preparation, (2) execution, and (3) completion, as shown on the left side of Figure 3.3. Our focus is on the UP procedure; hence, the CP procedures involving AMF and SMF are briefly explained, and the details can be found in [15].

Table 3.2: Xn-based handover scenarios in 5G and IUP deployments.

Xn-based Handover Scenarios	5G Deployment	**IUP Deployment**
Without UPF Reallocation	Indirect data forwarding from source UPF	x
With UPF Reallocation	Indirect data forwarding from source UPF	Direct data forwarding from source IUP

In the case of UPF reallocation for 5G deployment, the source gNB uses GTP-U tunnels over the Xn-U interface (cf. interface between gNB-1 and gNB-2 of Figure 3.1) to forward data from the source UPF to the target gNBs. Such indirect data forwarding will continue until the target UPF is applied as the new anchoring point (i.e., the N3 end-marker is used to identify the last forwarding packet). In contrast, the IUP deployment uses a peer-to-peer connection to forward packets since each IUP can act as the anchoring point, allowing IP packets to be forwarded directly between source and target IUPs. Moreover, because IUP already integrates UPF, there is no scenario without UPF reallocation. Finally, the CP procedure for IUP handover is similar to the one in [15], relying on N2 path switching done by AMF, as well as PDU session management, IP address allocation, and IUP selection handled by SMF. The SMF can mitigate service disruptions based on the supported Session and Service Continuity (SSC) mode [18], either preserving the existing IP address or assigning a new one as needed.

[4]There are further sub-categories within these two cases, but they have few variations in the UP procedure.

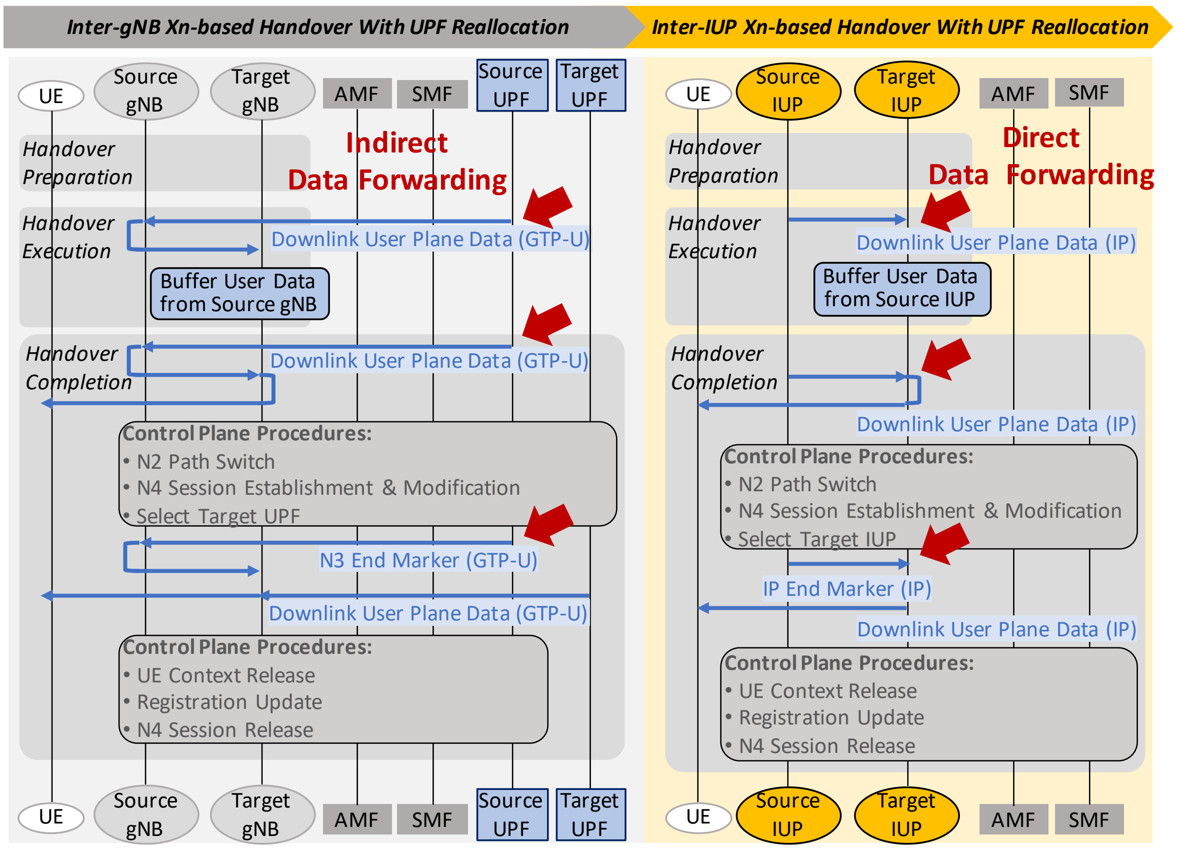


Figure 3.3: Xn-based handover with UPF re-allocation scenario in 5G and IUP deployments.

3.5.2 Roaming Architecture

In 5G home-routed roaming scenario [15], as shown in Figure 3.4, traffic is sent from an application server at Data Network (DN) in Home Public Land Mobile Network (HPLMN) to a UE in Visited PLMN (VPLMN). User data is traversed through UPF in HPLMN (H-UPF), UPF in VPLMN (V-UPF), gNB of VPLMN, and finally to UE. This procedure involves extra CP messages between network functions in HPLMN and VPLMN, e.g., SMF in HPLMN (H-SMF) and VPLMN (V-SMF) for session management via the N16 interface. Also, user data is forwarded between H-UPF and V-UPF over the N9 interface using GTP-U tunnels. These CP messages and UP forwarding add extra overhead and processing. For the local-breakout roaming scenario, user data is sent through the application server, UPF, gNB in VPLMN, making the procedure similar for both 5G and IUP deployments.

Conversely, IUP reduces these overheads, as shown in Figure 3.4, where the UE obtains its IP from the V-SMF and communicates directly with the application server in the HPLMN over IUP, bypassing H-UPF and V-UPF as well as eliminating the procedures between H-SMF and V-SMF. By serving as the default deployment, IUP simplifies traffic routing in roaming scenarios, making home-routed roaming similar to local-breakout roaming, with the only difference being the location of the application server while maintaining the respective traits: low latency for local-breakout roaming and better control for home-routed roaming.

Table 3.3: Roaming scenarios in 5G and IUP deployments.

Roaming Scenarios	5G Deployment	**IUP Deployment**
Home Routed	Indirect data transmission from DN in HPLMN	Direct data transmission from DN in VPLMN
Local breakout	Direct data transmission from DN in VPLMN	

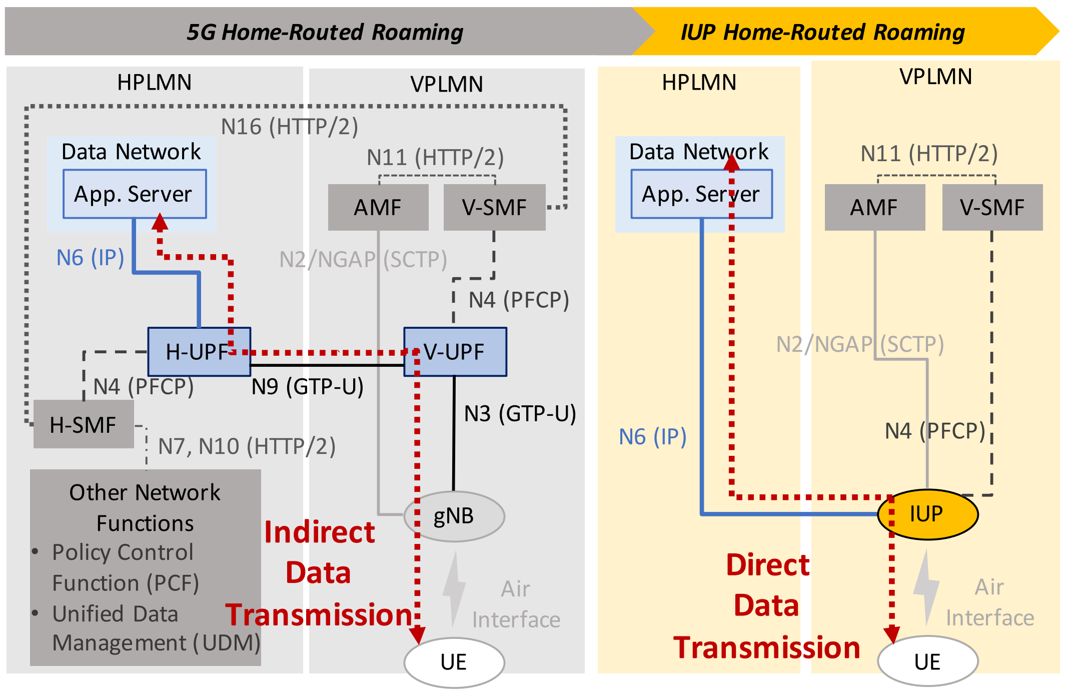


Figure 3.4: Home-Routed roaming scenario in 5G and IUP deployments.

3.5.3 RAN Disaggregation

The deployment of IUP aligns with the principles of Open RAN [127], supporting RAN disaggregation, where the RAN is divided into a CU and one or more DU(s) that communicate via the F1 interface. In 5G, the F1-U interface transmits user data over GTP-U tunnels (see gNB-CU-UP and gNB-DU in Figure 3.1), while the F1-C interface handles control signaling over SCTP. In the case of IUP, the F1-C interface continues to manage connections between CU-CP and DU, while IP protocol replaces GTU-U in the F1-U interface for user data transmission between CU-UP and DU (see IUP-CU-UP and IUP-DU in Figure 3.1). IUP enables user data flows and Downlink Data Delivery Status (DDDS) to be transmitted directly with an extra IP header, eliminating the need for GTP-U header between gNB-CU-UP and gNB-DU. Note that the IPSec protocol can be applied for encrypted transportation in between.

3.5.4 Non-3GPP Interworking

For interworking with non-3GPP networks, IUP can communicate directly with these networks (see Y2 interface between IUP-CU-UP and non-3GPP AP-2 of Figure 3.1), which is impossible in 5G deployment. This capability removes the need for additional network

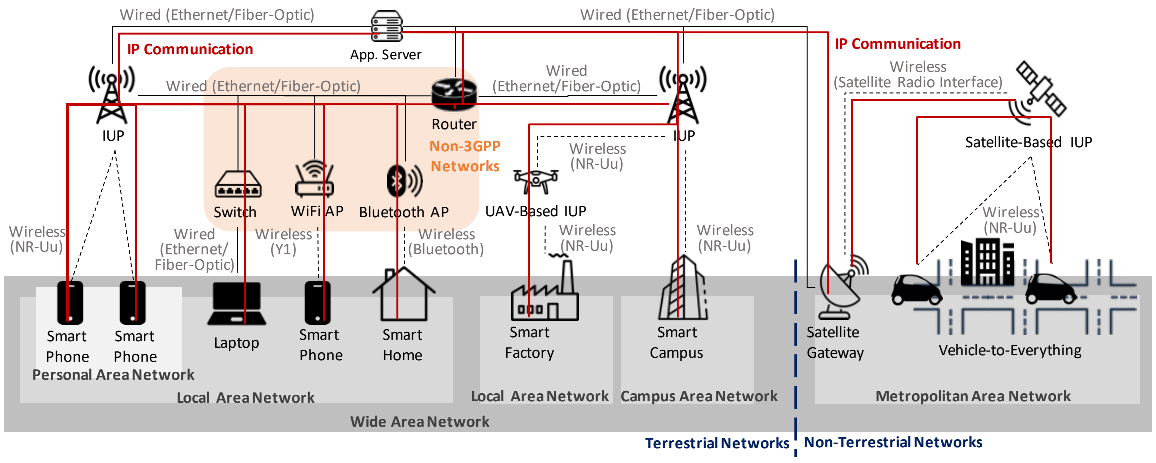


Figure 3.5: IUP interworking with various networks for universal connectivity.

functions, enabling real-time control across various access network technologies in the future. By eliminating the overhead associated with GTP-U tunnel processing and enhancing interoperability with non-3GPP networks, IUP not only reduces complexity but also improves scalability across various networks, as illustrated in Figure 3.5.

Moreover, IUP enables universal connectivity by functioning as a Layer 3 device, seamlessly integrating diverse wireless and wired technologies via the IP protocol without requiring additional network functions to bridge 3GPP and non-3GPP networks, as depicted in Figure 3.5. These networks range in scale from Personal Area Networks (PAN) to Local Area Networks (LAN), Campus Area Networks (CAN), Metropolitan Area Networks (MAN), and Wide Area Networks (WAN), supporting technologies like cellular, Ethernet, Wi-Fi, and Bluetooth across terrestrial, aerial, and non-terrestrial infrastructures.

3.5.5 Backward Compatibility

To maintain compatibility between IUP and 5G deployments, a key challenge is handling the interface between IUP and the existing UPF. Therefore, both GTP-U processing and SDAP sublayer are still required during the early deployment. Specifically, the GTP-U processing ensures that the existing UPF recognizes IUP as an I-UPF, utilizing the N9 interface between UPF and IUP. While in the event when the existing UPF would prefer to view IUP as a gNB via the N3 interface, the SDAP sublayer is needed to map incoming QoS flows into DRBs and the IDFC sublayer will be omitted, since the 3GPP-defined pipeline is done at the existing UPF. Once existing UPFs are upgraded, the additional GTP-U processing and SDAP sublayer in the IUP can be removed.

3.5.6 Collaborative Networking

In a collaborative network, a form of information-centric networking [27], multiple users need to seamlessly share data with each other to achieve a collective outcome whether as in perception (e.g., collaborative spatial computing) or content (e.g., media sharing, distributed caching). In particular, in collaborative spatial computing [109, 88] using VR, AR or XR, both physical and virtual environments are processed to render shared 3D objects, requiring reliable performance with minimal lag and maximum responsiveness. Meanwhile, in media sharing, the users transmit high-definition video streams, real-time audio, and interactive media with application in IP telephony and Closed-Circuit Television (CCTV) with less stringent latency requirements but high concerns for data locality, security, and privacy [132]. Finally, in distributed caching [125], strategical placement of data caches across the network is of interest to store frequently accessed content closer to end users, reducing the load on the external links of the network.

Current wireless technologies struggle to support the seamless interactions and immersive experiences demanded by the above applications due to the limitation of latency and bandwidth, as well as inefficient data processing and storage across distributed networks. Proposed solutions like Device-to-Device (D2D) communication and sidelink aim to enhance user experience by reducing reliance on centralized network infrastructure [145, 57]. However, ensuring sufficient reliability remains a challenge, especially when dealing with frequent exchanges and larger packets to accommodate richer sensor data and intended maneuvers.

3.5.7 Cloud Continuum

Cloud computing in 5G enables greater flexibility, scalability, and efficiency in network management while also supporting a diverse range of applications through edge computing [56, 94]. However, to meet the demands of emerging applications that require low latency or higher computational power and storage capacity, a unified and flexible infrastructure is necessary. This evolution is exemplified by offloading computing tasks closer to the endpoints, leveraging in-network computing, as highlighted by studies like [136]. The cloud continuum, integrating cloud computing resources seamlessly across various tiers of the mobile network from endpoints to the CN, is imperative for optimizing hardware and software resource allocation, as well as enhancing overall performance and user experience in the future mobile network systems.

3.6 Proof-of-Concept Implementation

In this section, we elaborate on how we integrate our proposed IUP architecture into a modern 5G mobile network, focusing on (1) GTP-U tunneling takeover, (2) programmable rules, and (3) a built-in routing mechanism. The IUP prototype is implemented based on OpenAirInterface (OAI) [114], an open-source software that provides an end-to-end 5G mobile network stack - from CN, RAN to UE. This implementation adds 7K lines of code, primarily to the RAN, with minimal modifications to the UPF. To validate the feasibility of IUP, we conducted experiments using commercial off-the-shelf UEs. Finally, we provide an overview of the hardware and software configurations used in our testbed.

3.6.1 Integrating IUP in a Contemporary 5G Mobile Network

GTP-U Tunneling Takeover. In 5G deployment, packets arriving at the N6 interface of UPF are processed through the 3GPP-defined packet processing pipeline [20], encapsulated with a GTP-U header, and sent to the RAN over the N3 interface via GTP-U tunnel(s). However, the job done by the GTP-U tunnel can be replaced with more efficient means such as direct function calls, memory mapping, or a Unix domain socket between UPF and RAN functionalities. In contrast, an IUP deployment integrates UPF functionalities directly into RAN, eliminating the need for a GTP-U tunnel and enabling direct data transmission from the N6 interface to the traffic management pipeline in IUP, allowing native IP flow handling within the RAN protocol stack. Avoiding packet encapsulation required by the GTP-U protocol accelerates the evolution of mobile networks, enabling faster communication compared to traditional network sockets [112]. Moreover, a transparent networking layer helps prevent common issues in higher layers, such as Head-of-Line (HOL) blocking, out-of-order delivery, or TCP meltdown, which can hinder flow control and retransmissions. These problems, inherent to encapsulation, significantly degrade the performance of applications demanding low latency or high throughput.

Programmable Functions and Rules. To enable the programmability of IUP, various approaches can be employed, including plug-in architectures, dynamic loading, and callbacks. IUP adopts a dynamic loading mechanism using shared libraries, allowing rules to be loaded and applied at runtime. This approach also supports plug-in architectures, enabling developers to seamleslly extend it. For the traffic management pipeline at the IDFC sublayer, an OSI-based PDRs are implemented in the classifier, which relies on information reported from layer-3 to layer-7 using the 5-tuple information in conjunction with a deep packet inspector, enabled through the open-source nDPI [79]. In the queuing stage, BARs are implemented with FIFO and CoDel [113]. The PSR in the scheduler uses a round-robin algorithm, while the TRR in the pacer is implemented with the 5G Bandwidth-Delay Product (5G-BDP) [91]. On the other hand, for the radio resource scheduler at the MAC sublayer, the DSR is implemented with Network Virtualization Substrate (NVS) [100], Early Deadline First (EDF) [86] and enhanced EDF (eEDF) algorithms, while the USR applies a proportional fair algorithm.

Built-in Routing Mechanism. By embedding IP processing within the RAN, IUP functions as a Layer 3 device, requiring a robust routing mechanism to manage various IP flows. To achieve this, IUP incorporates a BGP v4 agent [131], based on the BIRD Internet Routing Daemon [103], to advertise the routes for its served UEs within the designated network slices to other nodes. These nodes can be other IUPs, UPFs, or application servers running in the infrastructure, and the routes are advertised by peering the IUP to the other node agents. In a non-cloud-native environment, this peering can be done directly with other network instances (e.g., routers). However, in a cloud-native environment, the peering is made with the BGP agents on the cluster nodes based on the Kubernetes networking stack providers such as Calico or Cilium [140, 73]. Moreover, the Kubernetes networking provides the inter-container routing via BGP route advertisement. IUP extends its routing lists to include routes to the UEs and shares these routes with other containers in a similar manner to how inter-container routing is advertised. The main difference is that the IUP advertises routes directly to its own host node. For other nodes in the cluster, the IUP advertises the route with a “next-hop" parameter [131] set to its own host node, which is not supported by BGP v3 [59]. This ensures routing remains consistent with the existing infrastructure.

3.6.2 Testbed Configurations

Hardware. In our testbed, we have two setups: an IUP deployment and a 5G deployment. In the IUP deployment, we use four physical machines: machine 1 hosts the CN (excluding UPF), machine 2 hosts the IUP, and machine 3 and 4 are designated for UEs. The CN operates on machine 1, equipped with 8 CPUs and an Intel Core i5-8250U at 1.60GHz. The IUP runs on machine 2, which has 24 CPUs powered by an AMD Ryzen 9 5900X 12-Core processor at 3.7GHz and is connected to a USRP B210 SDR. For the UEs, we use Quectel RM500Q-GL modules, each connected to a separate machine: machine 3 with 8 CPUs and an Intel Core i7-8550U at 1.80GHz, and machine 4 with 40 CPUs and an Intel Xeon E5-2640 v4 at 2.40GHz. In the 5G deployment, the gNB runs on machine 2 (same as IUP in the previous setup), while the CN (including UPF) run on machine 1. Moreover, to test non-3GPP interworking, we set up a WiFi Access Point (AP) with 802.11g (WiFi generation 3), connected to a Google Pixel 5 smartphone.

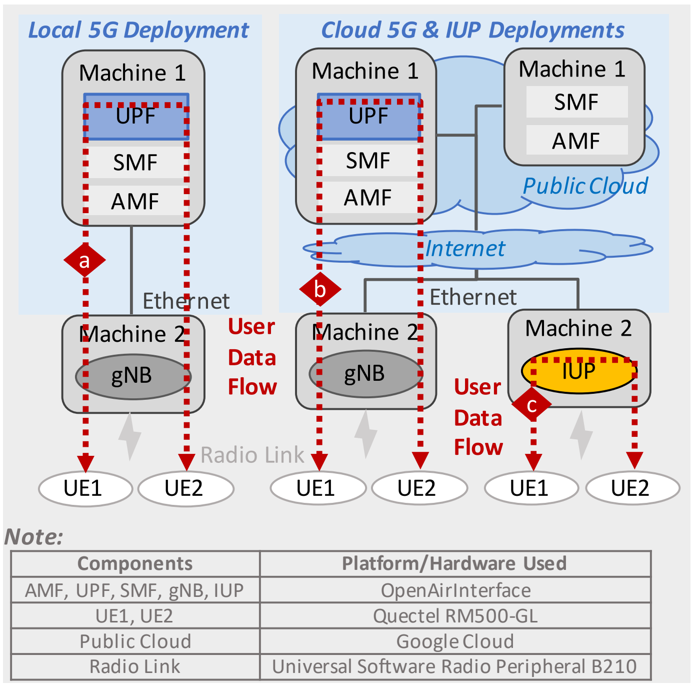
Software. The operating system for the machines running IUP and CN is Ubuntu 20.04 with Linux kernel 5.15.0. For the machines running Quectel UEs, it is Ubuntu 18.04 with Linux kernel 5.4.0. The IUP is implemented based on the latest release versions from OAI RAN v2.1.0 and OAI UPF v2.0.0. For CN, we used OAI CN v1.5.0, which includes AMF, SMF, and UDM. In the 5G deployment, OAI UPF v1.5.0 operates on the same machine as other CN components. Additionally, for the O-RAN framework, we used the NearRT-RIC and xApp from FlexRIC v2.0.0 [134, 70], which is an open-source and compliant with O-RAN specification. We developed our xApp based on FlexRIC to control the IUP through customized service models [64, 90]. Moreover, the clock of machines running UEs are synchronized with IUP and CN using the Network Time Protocol (NTP). For the radio configuration, IUP is set with 40 MHz bandwidth , subcarrier spacing 30 kHz and operates on a Time Division Duplex (TDD) pattern with 7 downlink slots, 2 uplink slots, and one mixed slot within every 5 ms frame. The maximum theoretical throughput is 175 Mbps, achieved with 256-QAM modulation and a single-input single-output (SISO) antenna configuration.

3.7 Experiment Evaluation

In this section, we present preliminary results by the over-the-air testbed, demonstrating how IUP can (1) reduce latency and overhead in UP performance, (2) seamlessly converged with non-3GPP network, and (3) provide programmability for IP flow traffic control and radio resource allocation.

3.7.1 Latency and Overhead Reduction

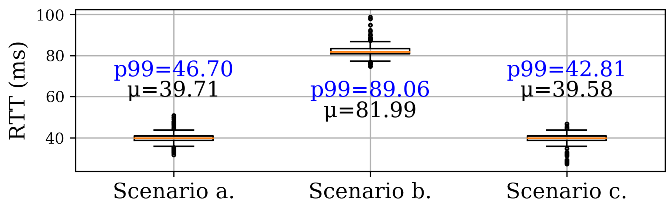
Setup and Workloads. As shown in Figure 3.6a, we measure the UP performance between two UEs in three scenarios: (a) Scenario a deploys the CN (including UPF) on a server next to the local gNB, creating a “short N3"; (b) Scenario b places the CN in a public cloud, connecting the local gNB via a “long N3"; (c) Scenario c deploys part of the CN (excluding UPF) in a public cloud with a local IUP, resulting in “no N3." In each scenario, we utilized a ping tool to measure the average RTT between two UEs connected to the gNB or IUP through an over-the-air testbed, with the results presented in Figure 3.6b. This experiment assumes local routing between UEs is enabled within both the UPF and IUP.



Note:

Components	Platform/Hardware Used
AMF, UPF, SMF, gNB, IUP	OpenAirInterface
UE1, UE2	Quectel RM500-GL
Public Cloud	Google Cloud
Radio Link	Universal Software Radio Peripheral B210

(a) Experiment setup.



(b) Measured RTT between two UEs.

Figure 3.6: Latency measurement in local 5G, cloud 5G, and cloud IUP scenarios.

5G vs. IUP Deployments. While 3GPP provides several deployment options to support low-latency scenarios, such as placing UPF closer to the edge, the fundamental network delays still persist. As an example, Figure 3.6b compares latency when deploying UPF locally near gNB (see Scenario a) versus deploying UPF in a public cloud (see Scenario b). The

Table 3.4: Protocol header overhead in VoIP and UDP for Legacy and IUP deployments.

Scenarios			User Packet			Overhead
				IP Packet		
Use Cases	Network Deployment Between RAN and UPF		GTP Header	IP Header	User Data	(Headers % of User Packet)
IPv4 VoIP	Legacy	IPv4	44 bytes	40 bytes	31 bytes	73.0%
		IPv6	64 bytes			77.0%
	IUP	No physical separation and network transmission	0 bytes			56.3%
IPv4 TCP	Legacy	IPv4	44 bytes	40 bytes	1416 bytes	5.6%
		IPv6	64 bytes		1396 bytes	6.9%
	IUP	No physical separation and network transmission	0 bytes		1460 bytes	2.6%

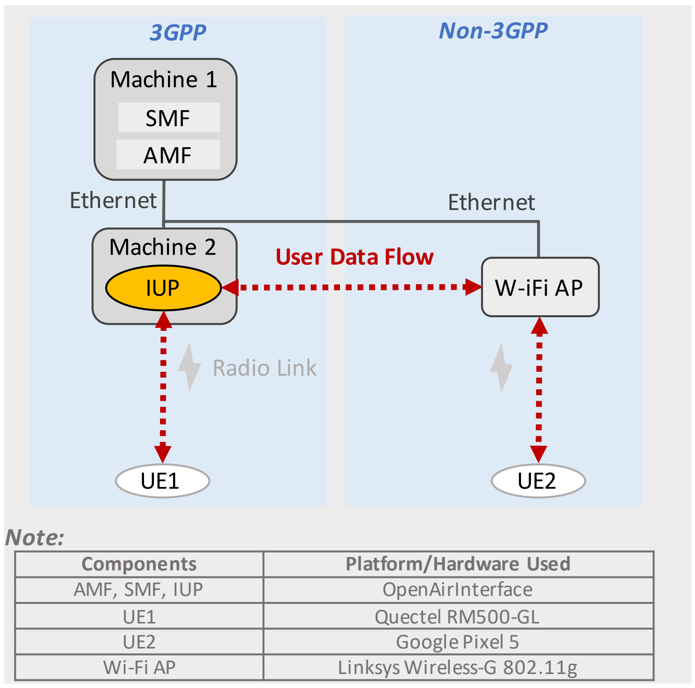
average RTT between two UEs significant increases when UPF is located farther from gNB, confirming that physical proximity of endpoint devices does not necessarily ensure low-latency network connections, as additional hops introduce notable delays. IUP addresses this by offering an integrating solution that reduces the network hops to a minimum of one. As shown by Scenario c in Figure 3.6b, UP data only traverses within IUP, resulting in a lower RTT of 39.58 ms compared to Scenario a, which experiences non-negligible network delays between UPF and gNB. The improvement is also seen in P99 latency, due to the removal of N3 interface.

Efficient Data Delivery. Moreover, IUP removes the GTP-U protocol processing and its associated headers in each packet, including outer IP, UDP, and GTP-U headers. This enables user data to be delivered efficiently without requiring a smaller MTU size to avoid IP fragmentation [141]. Specifically, for IPv4 and IPv6 protocols, these headers consume 44 bytes and 64 bytes, respectively. For instance, when delivering a 60-byte G.729 VoIP packet, the extra GTP header would consume up to 64 bytes, resulting in a total packet size of 124 bytes. In this case, IUP removes the need for a GTP header, efficiently delivering the 60-byte packet and reducing overhead by 50%, as depicted in Table 3.4.

3.7.2 Converged with Non-3GPP Network

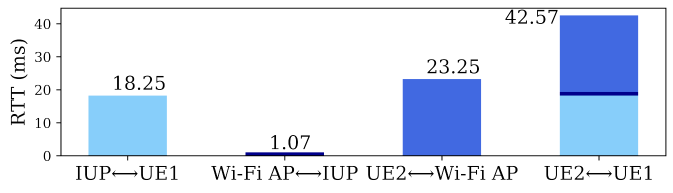
Setup and Workloads. To showcase the benefits of IUP, we set up a scenario with UE1 connected to IUP and UE2 connected to a Wi-Fi AP, as shown in Figure 3.7a. This setup demonstrates that IUP can provide connectivity even when one UE is served by a non-3GPP network (compared to Scenario c in Figure 3.6a, where IUP serves both UEs). We utilized ping tool to measure the average RTT between two UEs, with detailed latency breakdown results presented in Figure 3.7b. Additionally, we used VLC media player to transfer a live MPEG4 video via UDP from UE1 to UE2.

Direct Communication via IP. As discussed before, IUP can be converged with non-3GPP networks without the need for extra network functions. As illustrated in Figure 3.7a,



Components	Platform/Hardware Used
AMF, SMF, IUP	OpenAirInterface
UE1	Quectel RM500-GL
UE2	Google Pixel 5
Wi-Fi AP	Linksys Wireless-G 802.11g

(a) Experiment setup.



(b) Measured RTT between two UEs.

Figure 3.7: Scenario that two UEs served by IUP and Wi-Fi AP respectively.

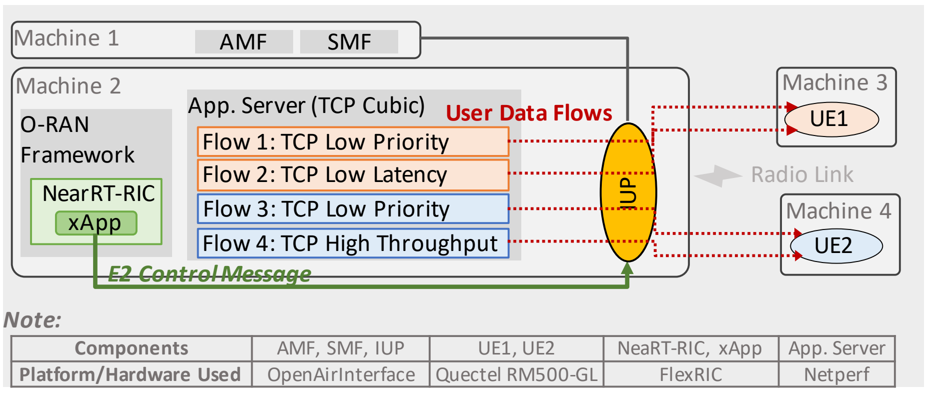
we set up a scenario where UE1 connects to an IUP and UE2 connects to a Wi-Fi AP, with an Ethernet connection between IUP and Wi-Fi AP. In this sense, IUP works as a Layer 3 router, facilitating direct communication with Wi-Fi AP via IP protocol and shortening the UP path between UE1 and UE2 by avoiding the need to route user data to the CN.

Latency Breakdown. The average RTT between UE1 and UE2 is approximately 42.57 ms, as shown in Figure 3.7b, which is slightly larger than the result in Scenario c of Figure 3.6b. Additionally, we break down the average RTT between UE1 and UE2 into three components. First, the RTT between IUP and Wi-Fi AP over Ethernet is negligible, whereas the major components are the air-interface delays between IUP and UE1 as well as between Wi-Fi AP and UE2. Additionally, the average RTT between IUP and UE1 is about 5 ms lower than that between Wi-Fi AP and UE2. This difference might due to the contention window in the Wi-Fi standard, which requires competition for transmission opportunities, unlike the scheduling-based approach in 3GPP.

Peer-to-Peer Video Streaming. Additionally, to validate the feasibility of this setup, we successfully streamed a live video from UE1 to UE2 at 720 p resolution with a bitrate of 2119 kbps[5]. The video was transmitted via the uplink channel of IUP, configured with a TDD frame structure of 7 downlink slots, 2 uplink slots, and 1 mixed slot. As a result, the uplink bandwidth was lower compared to the downlink.

3.7.3 Programmability over IP Flows and Radio Resources

Setup and Workloads. To demonstrate the programmability of IUP, a new xApp is designed to control both IP flows and radio resources, as illustrated in Figure 3.8. This xApp enforces rules for the traffic management pipeline at the IDFC sublayer and the radio resource scheduler at the MAC sublayer. For the traffic management pipeline, we modify PDRs, BARs, and QRRs to adjust the classifier, queuing, and shaper, respectively. For the radio resource scheduler, the scheduling policy is adjusted according to the applied USRs, including the maximum scheduling rate and scheduling deadline. Moreover, in our experiment setup, two UEs are connected to the IUP. Each UE receives two TCP traffic flows, generated using Netperf, with different Differentiated Services Code Point (DSCP) values in the IP header, which are mapped to the same DRB, i.e., only one DRB per UE. Specifically, Flow 2 on UE1 requires low-latency service (DSCP: AF11) with lower bandwidth, while Flow 4 on UE2 demands higher throughput (DSCP: AF21); Flow 1 on UE1 and Flow 3 on UE2 are low-priority services (DSCP: CS1). Latency for each flow is measured using fping, which sends 84 bytes ICMP packet every 200 ms. Additionally, IUP operates the RLC in Acknowledgment Mode (AM), with the TCP congestion control algorithm set to Cubic for the machine, running the Netperf clients. Then, we modify the rules in four different scenarios (see detailed of applied rules in Table 3.5) and measure network statistics such as Resource Block (RB) usage and RLC queue size, as well as application performance like data rate and end-to-end latency of each flow.



Components	AMF, SMF, IUP	UE1, UE2	NeaRT-RIC, xApp	App. Server
Platform/Hardware Used	OpenAirInterface	Quectel RM500-GL	FlexRIC	Netperf

Figure 3.8: Experiment setup for scenarios with xApp on NearRT-RIC controlling IUP.

Radio Resource Control. At the start of our experiment (from 0 s to 60 s in Figure 3.9 and Figure 3.10), the xApp is not configured, and a fair share of radio resource is allocated

[5]Demo video available at:https://youtu.be/SsEyCUoTc5M

Table 3.5: Applied rules in each scenario.

Scenarios	Applied Rules				
	Radio Resource Scheduler		Traffic Management Pipeline		
	UE	USR	Classifier	Queuing	Shaper
1	UE1	Bitrate: 50 Mbps	x		
	UE2	Bitrate: 110 Mbps			
2	UE1	Bitrate: 50 Mbps	x		
		Deadline: 2.5 ms			
	UE2	Bitrate: 110 Mbps			
		Deadline: 5 ms			
3	UE1	same rules as previous scenario	PDR1: Flow1	BAR1: CoDel	QRR1: 10 Mpbs
			PDR2: Flow2	BAR2: CoDel	QRR2: 50 Mpbs
	UE2		PDR3: Flow3	BAR3: CoDel	QRR3: 10 Mpbs
			PDR4: Flow4	BAR4: FIFO	QRR4: 110 Mpbs
4	UE1	Bitrate: 50 Mbps	same rules as previous scenario		QRR1: 10 Mpbs
		Deadline: 2.5 ms			QRR2: 30 Mpbs
	UE2	Bitrate: 120 Mbps			QRR3: 10 Mpbs
		Deadline: 5 ms			QRR4: 120 Mpbs

to both UEs to make each flow share the same rate (∼40 Mbps). The xApp then applies the USR (Scenario 1 in Table 3.5) to the MAC radio resource scheduler at 60 s (Scenario 1), where radio resources are scheduled proportionately based on the defined maximum scheduling rates. Thus, each flow of UE1 and UE2 achieves throughput of 25 Mbps and 55 Mbps, respectively. Note that such proportional scheduling approach does not consider any scheduling deadline; therefore, all available resources are allocated to either UE per time slot, leaving few unused RBs. In comparison, when scheduling deadlines are introduced in Scenario 2 (from 120 s to 180 s in Figure 3.9), radio resources of each slot are allocated accordingly, resulting in more unused RBs. This difference arises because the USR applied in Scenario 2 dynamically adjusts radio resources based on the required bitrate, deadline, and channel quality of the scheduled UE, ensuring efficient resource allocation without over-provisioning (see [64] for details on the EDF scheduling algorithm).

It is evident that controlling only the radio resource allocation does not differentiate application performance, e.g., flows 1 and 2 of UE1 as well as flows 3 and 4 of UE2 have the same data rate and end-to-end latency. Although the USR applied to each UE results in different latency for UE1 and UE2, flow 2 still experiences high RLC queue sizes, as depicted in Figure 3.10, causing a higher latency (∼1000 ms) that does not meet its low-latency requirement. In the next two scenarios, the traffic management pipeline at the IDFC sublayer is also controlled to address this issue.

IP Flow Control. To differentiate IP flows within each DRB, the xApp applies PDRs, BARs, and QRRs along with the previous USR in Scenario 3 (from 180 s to 240 s in Figure 3.9 and Figure 3.10). Specifically, as shown in Scenario 3 of Table 3.5, these PDRs classify each flow into individual queues, each of which is controlled by a respective BAR. For instance, flow 2 uses CoDel active queue management for lower latency, configured with a target delay of 50 ms[6], while flow 4 utilizes a FIFO queue to maximize throughput. Both flows 1 and 3 are assigned to individual CoDel queues with a relaxed target delay of 100 ms This

[6]RFC 8289 recommends a target delay of 5-10% of RTT [113].

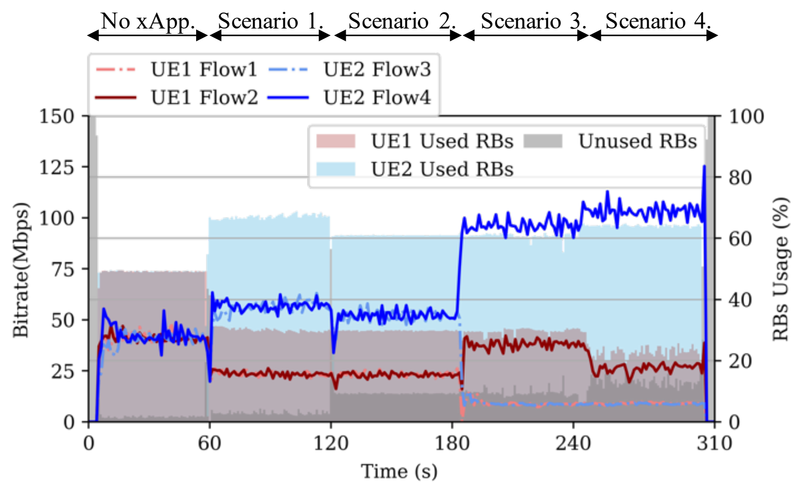


Figure 3.9: Measured application bitrate and radio resource usage in different scenarios.

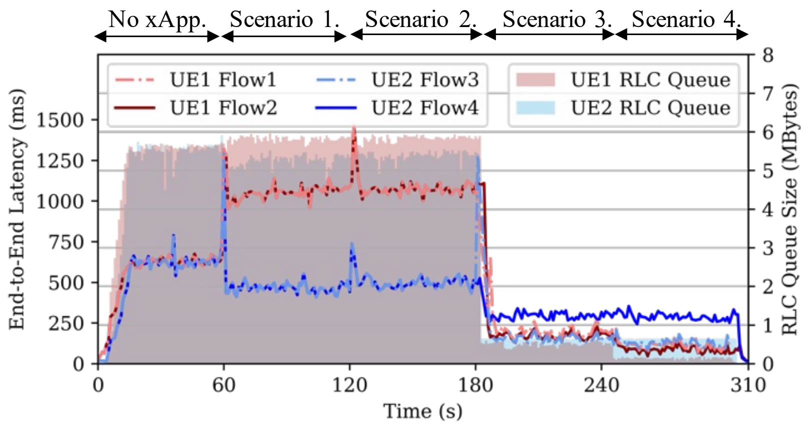


Figure 3.10: Measured application latency and network queuing size in different scenarios.

configuration helps prevent excessive queue sizes (∼3 Mbytes) and delays (∼2 s) that may occur with FIFO queues. Afterwards, QRRs limit the egress rate of each queue, and a default round-robin scheduler is used in PSR to schedule packets across queues. As shown in the results from Figure 3.9 and Figure 3.10, each flow now behaves differently. First, due to the applied QRRs, flows 1 and 3 have lower data rates, while flows 2 and 4 have higher data rates. Additionally, CoDel makes a major contribution to reducing the end-to-end latency of flow 2 by over 80%. Furthermore, the RLC queue size for both UEs decreases significantly due to the default 5G-BDP pacer [90] applied to TRR, while the RB allocation remains unchanged due to the same USRs.

Coordinated Control. In the final scenario (from 240 s to 300 s in Figure 3.9 and Figure 3.10), both USRs and QRRs are updated to coordinate traffic control and resource allocation. We observe that the updated QRR reduces the end-to-end latency of flow 2. This adjustment lowers the egress rate of flow 2 to ensure that the overall egress rates for UE1 (i.e., QRR1 and QRR2) remain within the maximum scheduling rate specified by the corresponding USR. Also, both the egress rate of flow 4 and the maximum scheduling rate for UE2 are increased, allowing for more radio resources to be allocated to flow 4. Note that even more RBs are unused, but we can better satisfy the needs of user data flows, i.e., lower latency for flow 2 and higher throughput for flow 4.

In summary, IP traffic control and radio resource allocation are equally essential to meet the diverse needs of applications. IUP provides programmability from both perspectives, making it a versatile solution for a wide range of use cases.

3.8 Discussion and Future Work

5G aims to deliver higher data rates, lower access latency, and greater deployment flexibility compared to prior generations. This evolution expands business models (e.g., neutral hosts [116], vertical industries) and creates new use cases (e.g., private 5G, Industry 4.0). However, the full potential of 5G has not yet be fully realized [80]. One reason is that the control options in 5G are too complex for application developers to use effectively, unlike the quick process of releasing new software versions. For example, several 5G QoS Indicators (5QIs) [18] are standardized from a mobile network viewpoint, but applications often operate on a best-effort basis due to limited visibility into their performance impact within network. On the other hand, applications prefer network to act as simple data pipe(s), focusing on efficient packet forwarding and routing for their traffic flows, as they are already resilient to IP address changes and capable of handling multiple connections.

Therefore, in this chapter, the concept of IUP introduces IP flow processing into RAN at the access side, unifying the traditionally separate flow management functions of UPF and RAN. This advancement streamlines mobile network infrastructure and enables seamless interworking with non-3GPP networks. However, embedding UPF within RAN presents several challenges that must be carefully addressed.

One major concern is the limited understanding of the additional processing overhead introduced when embedding the traffic management pipeline, or even parts of it, within the RAN. A key concern is scalability, as its per-flow operations may lead to computational overhead. However, unlike conventional centralized UPFs that manage traffic across multiple cells, IUP operates at the per-cell level, significantly reducing scalability demands. While implementation challenges exist, they can be mitigated through optimized techniques (e.g., tree-based structures) and hardware acceleration (e.g., SmartNICs).

Our proof-of-concept implementation demonstrates that deploying IUP increases memory usage to approximately 4.6%, compared to 4.2% in a standard 5G deployment. Similarly, CPU consumption rises to 66.9% for IUP, compared to 55.1% for a 5G deployment encompassing both RAN and UPF. This increase stems from the overhead introduced by programmable functions and rules implemented through shared libraries, as well as the use of Unix domain sockets, which incur non-negligible CPU and memory costs [112]. The extent of this additional resource consumption is closely tied to the implementation methods used for IUP, suggesting that it can be optimized by exploring alternative approaches.

Beyond computational efficiency, a trade-off analysis between the IUP deployment and the current N-to-1 RAN-UPF design is required to verify both cost and energy efficiencies in a scalable IUP deployment. Additionally, the mapping from IP flows to DRBs is still an open issue, which was done jointly by UPF and RAN in the past, but is now done only on IUP. As 3GPP and non-3GPP networks converge, it is critical to analyze the present QoS framework and determine how it might be applied to non-3GPP link technology. Finally, further research is needed to explore how the orchestration and management can enable xApp programmability, allowing the network to automatically adjust rules based on real-time observations to enhance IUP's adaptability and performance.

3.9 Conclusions

In this chapter, we introduce IUP for next-generation mobile networks that integrates UPF functionalities into the RAN. It provides programmability through a new traffic management pipeline in the IDFC sublayer, as well as multiple programmable rules for IP traffic control and radio resource allocation. Key benefits of IUP include reduced latency and overhead, enhanced programmability of the UP, simplified CP operations, and seamless interworking with non-3GPP networks. Additionally, several key use cases are analyzed to address how IUP operates in CP procedures, such as handover and roaming, and how it is compatible with existing deployments. Finally, our real-world testbed results highlight several benefits of IUP, including reduced network latency and overhead, seamless convergence between 3GPP and non-3GPP networks, and programmability across IP traffic control and radio resource allocation to serve different applications.

Chapter 4

Autonomous Radio Access Network

4.1 Introduction

The modern mobile network infrastructure relies not only on 3GPP-defined architectures but also incorporates O-RAN architectures to enhance the programmability and interoperability of the Radio Access Network (RAN) while addressing the growing demands of emerging technologies, such as Augmented Reality (AR) or Virtual Reality (VR). However, these two ecosystems contribute to the mobile network from different perspectives, as illustrated in Figure 4.1, , potentially posing challenges for network operators in managing and integrating both systems.

As depicted by the green blocks in Figure 4.1, 3GPP establishes the global foundation for mobile networks by developing standardized protocols that ensure seamless connectivity and global interoperability [1]. Additionally, it provides specifications for network operators to manage and optimize their networks, as well as generic design and study reports for emerging use cases. In contrast, as depicted by the blue blocks in Figure 4.1, O-RAN focuses specifically on the RAN, introducing open interfaces to enable interoperability between RAN components from multiple vendors while relying on 3GPP-defined architectures.

By fostering openness and vendor neutrality, O-RAN empowers network operators with greater adaptability and reduced vendor dependency. Moreover, it allows the dynamic programming of RAN functionalities using advanced technologies such as Artificial Intelligence (AI) and Machine Learning (ML) models, further enhancing flexibility for operators to optimize RAN performance. However, this enhancement also introduces additional overhead and complexities to the network, increasing the burden on network operators.

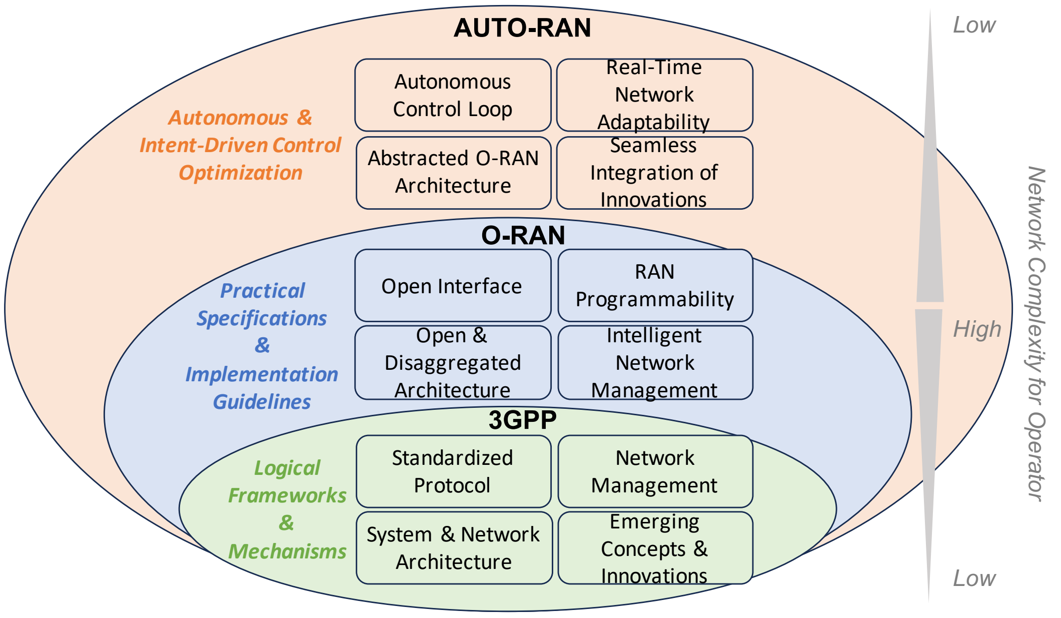

AUTO-RAN
Autonomous & Intent-Driven Control Optimization
Autonomous Control Loop
Real-Time Network Adaptability
Abstracted O-RAN Architecture
Seamless Integration of Innovations
O-RAN
Practical Specifications & Implementation Guidelines
Open Interface
RAN Programmability
Open & Disaggregated Architecture
Intelligent Network Management
3GPP
Logical Frameworks & Mechanisms
Standardized Protocol
Network Management
System & Network Architecture
Emerging Concepts & Innovations
Low
High
Low
Network Complexity for Operator

Figure 4.1: Overview of 3GPP, O-RAN and AUTO-RAN contributions in mobile networks.

[1]This allows endpoint devices from various manufacturers to operate across 5G networks provided by different operators and vendors.

Challenges. The coexistence of 3GPP and O-RAN ecosystems presents significant challenges for mobile network operators. While 3GPP provides globally standardized architectures, O-RAN introduces additional components, such as the RAN Intelligent Controller (RIC), to enable programmability and real-time optimization. Integrating these components into existing 3GPP-based systems increases operational complexity, particularly due to limited coordination between the two ecosystems. Additionally, O-RAN technologies often fall outside the scope of 3GPP standards, requiring operators to adopt and manage new frameworks and tools. This dual-system approach imposes technical and operational burdens, especially on operators not mandated to implement O-RAN architectures, and complicates end-to-end network optimization.

Contributions. In this chapter, we introduce Autonomous Radio Access Network (AUTO-RAN), illustrated by the orange blocks in Figure 4.1, a novel concept that enhances the current O-RAN architecture with an autonomous control loop, simplifying the network optimization process for operators controlling infrastructures that integrate both 3GPP and O-RAN ecosystems. AUTO-RAN abstracts the underlying complexities of O-RAN, enabling operators to interact with the network using familiar 3GPP-defined logic, while leveraging O-RAN technologies to enable real-time adaptability and performance optimization. It also facilitates the smooth integration of new features introduced by either 3GPP or O-RAN, offering a future-proof foundation that bridges both architectures. We detail the motivation behind AUTO-RAN (Section 4.2), describe its architecture and autonomous control loop (Section 4.3), and present a proof-of-concept implementation, including examples of xApps and rApps that provide a declarative mechanism for operator interaction (Section 4.4.3). Finally, we demonstrate two key use cases — automated RAN slicing and mobility management — showcasing AUTO-RAN's real-time adaptability to improve both network and user performance, while abstracting control complexity for network operators (Section 4.5).

4.2 Why AUTO-RAN

In this section, we highlight the role of AUTO-RAN by addressing the challenges network operators face with dual ecosystems — 3GPP and O-RAN. These include increased complexity of mobile networks, limited real-time adaptability, and difficulties in integrating new features. The following subsections provide details on how AUTO-RAN effectively tackles these challenges.

4.2.1 Reduced Complexity for Operators in Dual Ecosystems

In 5G, the mobile network landscape is defined by two major ecosystems: 3GPP and O-RAN, as illustrated in Figure 4.2. While both aim to optimize network and service performance, they adopt fundamentally different approaches. 3GPP provides a comprehensive framework for end-to-end network operations, ensuring global interoperability and seamless connectivity for endpoint devices. This enables operators to optimize network and service performance from an end-to-end perspective, as illustrated in the right side of Fig-

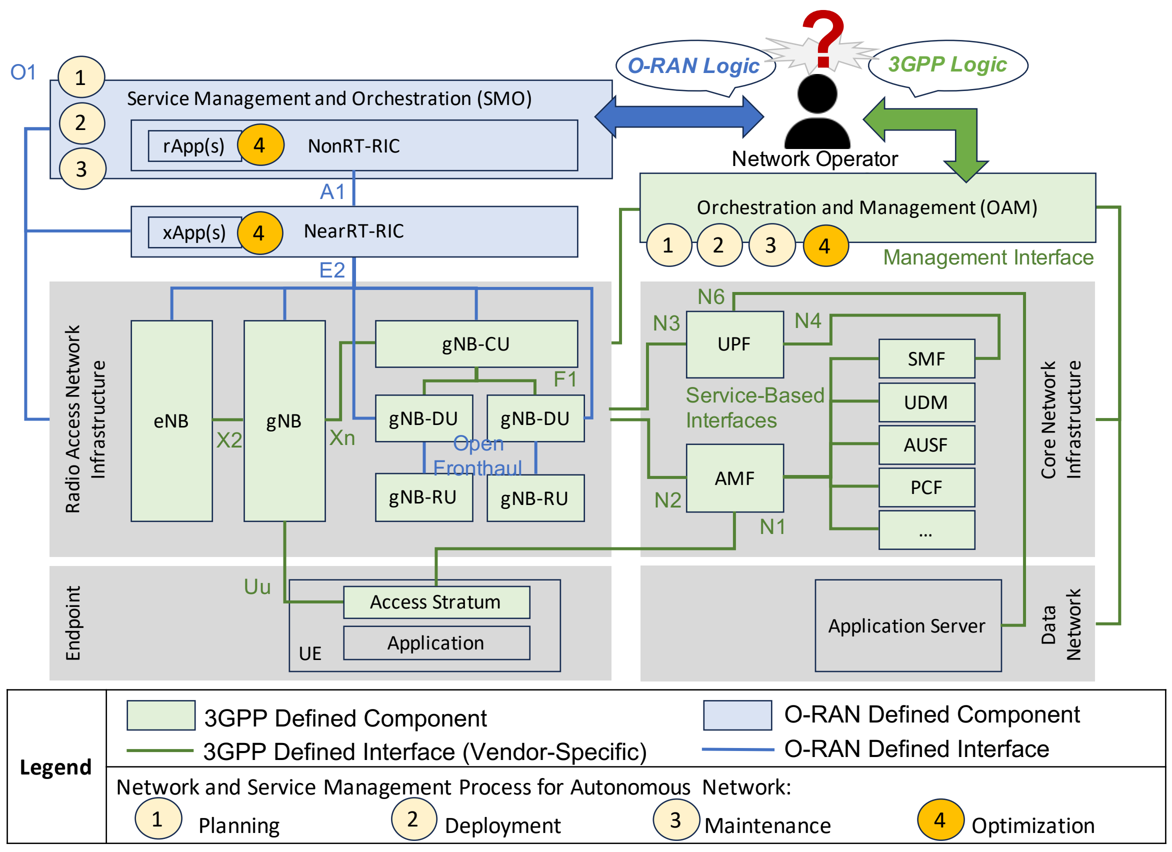


Figure 4.2: Increased complexity for network operators in managing autonomous networks across two ecosystems.

ure 4.2. In contrast, O-RAN focuses specifically on RAN programmability, introducing open interfaces and software-driven flexibility. This approach allows network operators to optimize network and service performance in real-time and within multi-vendor environments, as illustrated in the left side of Figure 4.2. However, the coexistence of these two ecosystems adds complexity for network operators, who must manage and integrate two distinct operational logics. The following section delves into these ecosystems in greater detail.

4.2.1.1 3GPP

As a key organization in global telecommunications, 3GPP develops comprehensive technical specifications (TS) that define protocols, radio technologies and architectures, as summarized in Table 4.1 These protocols include examples such as Medium Access Control (MAC) [10], Radio Link Control (RLC) [13], Packet Data Convergence Protocol (PDCP), among others. Beyond protocols, 3GPP defines the foundational system and network architectures, including the 5G Core Network (CN) [18] and RAN [8], as shown in the right side of Figure 4.2. Furthermore, it provides frameworks for network management and optimization, such as Self-Organizing Networks (SON) [7], Management and Orchestration (OAM) [4], which are essential for efficient and flexible network operations.

In parallel, 3GPP provides technical reports (TR) to explore and evaluate emerging technologies and trends. These include the integration of Artificial Intelligence (AI) and Machine Learning (ML) in mobile networks [17, 16], the definition of Key Quality Indicators (KQIs) for assessing 5G service experience [24], advancements in digital twins [22], and the development of Non-Terrestrial Networks (NTN) [25], among other innovative areas.

Table 4.1: Overview of 3GPP contributions.

Category	Description (Example)	Technical Specification/Report
Standardized Protocol	Medium Access Control Protocol	TS 38.321 [10]
	Radio Link Control Protocol	TS 38.322 [13]
	Packet Data Convergence Protocol	TS 38.323 [12]
	Radio Resource Control Protocol	TS 38.331 [14]
	Session Initiation Protocol	TS 24.229 [19]
System and Network Architecture	5G Core Network	TS 23.501 [18]
	5G Radio Access Network	TS 38.401 [8]
	IP Multimedia Subsystem	TS 23.228 [9]
Network Management and Optimization	Self-Organizing Networks	TS 28.313 [7]
	Management and Orchestration	TS 28.533 [4]
	Network Slicing	TS 28.530 [5]
	Autonomous Networks	TS 28.100 [6]
	Management Control Loops	TS 28.535 [21]
Emerging Concepts and Innovations	Artificial Intelligence and Machine Learning	TR 38.743 [17], TR 38.908 [16]
	Key Quality Indicators	TR 28.863 [24]
	Digital Twins	TR 28.915 [22]
	Non-Terrestrial Networks	TR 28.874 [25]

3GPP defines "*logical frameworks and mechanisms*" to provide a standardized foundation for implementing advanced network concepts while avoiding prescriptive implementation details. This flexibility allows vendors to design innovative solutions and enables operators to deploy customized implementations tailored to their network environments. For example, 3GPP introduces the concept of management control loops for network optimization but leaves the specifics of decision-making, such as enabling network slicing, traffic steering, or load balancing, to vendors or network operators.

However, this approach poses challenges for operators managing heterogeneous network functions from multiple vendors. A lack of visibility into proprietary vendor mechanisms often leads to vendor lock-in, making it difficult for operators to integrate and manage multi-vendor networks. Moreover, as 5G evolves, the increasing number of features (e.g., dynamic spectrum sharing, beamforming), applications (e.g., AR/VR), and new use cases (e.g., NTN) further compounds the complexity. To address these vendor-specific challenges, O-RAN emerges as a solution, which we explore further in the next subsection.

4.2.1.2 O-RAN

The O-RAN Alliance [36] was established to address the challenges posed by traditional, proprietary mobile network architectures (e.g., vendor lock-in and limited interoperability). By focusing on the RAN domain, O-RAN provides TS and TR to define the open interfaces and disaggregated architectures based on the 3GPP specifications, enabling interoperability, intelligent network optimization and programmability, as summarized in Table 4.2

Table 4.2: Overview of O-RAN contributions.

Category	Description (Example)	Technical Specification/Report
Open Interface	A1	Working Group 2 [29]
	E2	Working Group 3 [42]
	O1	Working Group 10 [39]
	Open F1,W1,E1,X2,Xn	Working Group 5 [32, 47]
	Open Fronthaul (Split 7-2x)	Working Group 4 [48]
Open and Disaggregated Architecture	O-RAN Architecture	Working Group 1 [117]
	Open X-Haul	Working Group 9 [33]
	White-box Hardware	Working Group 7 [34, 35]
	Interoperability Test	Working Group 4 & 8 [31, 49]
Intelligent Network Optimization	rApp & Non Real-Time RAN Intelligent Controller	Working Group 2 [30]
	xApp & Near Real-Time RAN Intelligent Controller	Working Group 3 [40]
	Operations and Maintenance	Working Group 10 [38]
	Intents-Driven Management	Working Group 1 [37]
RAN Programmability	RAN Control Service Model	Working Group 3 [46]
	Cell Configuration and Control Service Model	Working Group 3 [44]

Key interfaces such as A1 [29], E2 [42], and O1 [39] play vital roles in the O-RAN architecture, as illustrated in the left side of Figure 4.2, by enabling centralized control and management across E2-Nodes (denoted as RAN components in 3GPP terminology, such as eNB, gNB, gNB-CU, or gNB-DU). Additionally, the open fronthaul [48] enables interoperability between DU and RU, while the open F1, E1, and Xn concepts [32, 47] promoted by O-RAN build upon 3GPP specifications to enable multi-vendor interoperability between disaggregated RAN components (E2-Nodes). Based on these open interfaces, O-RAN introduces a fully open and disaggregated architecture [117] that not only focuses on communication interfaces between/across E2-Nodes but also extends to the transport domain (e.g., open x-haul [33]). Furthermore, O-RAN enables decoupling of software and hardware platforms with solutions such as white-box hardware [34, 35], while ensuring seamless multi-vendor integration through interoperability test specifications [49, 31].

Moreover, a key innovation in O-RAN is the RAN Intelligent Controllers, which includes the NonRT-RIC [30] and NearRT-RIC [40], as shown in Figure 4.2. O-RAN introduces components such as rApps on the NonRT-RIC and xApps on the NearRT-RIC, which can integrate advanced functionalities (e.g., AI/ML models) to bring intelligence into mobile networks. These components optimize network performance and enable emerging capabilities, such as intent-driven management [37]. Inspired by the SDN concept, the RIC enables dynamic programmability of the RAN at runtime. This programmability is facilitated through O-RAN-defined Service Models (SM), such as the RAN Control (RC) [46], the Cell Configuration and Control (CCC) [44] and the Low Layers Control (LLC) [45]. These models standardize the structure of control messages exchanged between the SDN controllers (e.g., NonRT-RIC, NearRT-RIC) and the controlled entities (e.g., E2-Nodes). As a result, control applications (e.g., xApps and rApps) can manage various RAN functions (e.g., MAC scheduler) across E2-Nodes from different vendors, enabling a unified and interoperable control framework.

O-RAN defines“*practical specifications and implementation guidelines*” to enable open, disaggregated, and multi-vendor RAN solutions that dynamically optimize network performance, while preserving the functionalities and architecture defined by 3GPP. For example, while 3GPP define the QoS framework for optimizing the service performance, O-RAN enables vendors to develop and provide xApps implemented with RC SM. These xApps allow operators to dynamically perform QoS management (e.g., radio bearer control and radio resource allocation control) in near-real-time (with control loop latency of 10 ms to 1 s). Importantly, these optimizations can be applied directly within the RAN, often without requiring extensive network-wide changes or reconfiguration.

However, this approach focuses solely on RAN optimization, which creates inconsistencies in achieving a cohesive end-to-end network perspective. It lacks explicit provisions for managing the integration of CN and RAN, placing this burden on the network operators. This, in turn, increases complexity for network operators as they must navigate and manage two ecosystems. To address this, we introduce the concept of AUTO-RAN in this chapter, as illustrated in Figure 4.3. AUTO-RAN bridges these two ecosystems, enabling seamless integration and facilitating end-to-end network and service optimization. This approach delivers a unified and streamlined solution, reducing operational challenges for network operators. Further details are elaborated in Section 4.3.

4.2.2 Enabled Real-Time Network Adaptability

Real-time network adaptability has become increasingly crucial in mobile networks dues to the complex and dynamic nature of modern network conditions. Factors such as fluctuating channel quality and the increasing diversity of applications running simultaneously on user devices challenge traditional service-based network optimization approaches, which rely on broad traffic categories (e.g., voice, messaging, or data). In modern use cases, users often run multiple high-demand applications simultaneously, which may be grouped into the same category, leading to resource contention and performance degradation. For example, live video streaming of a sports event and video calls with friends often compete for shared resources, as network operators typically allocate only one DRB for data service.

This shift from predictable service-level traffic to dynamic application-level demands introduces variability and unpredictability in network performance. Current approaches, such as manual or semi-automated QoS reconfigurations, are reactive, costly, and struggle to keep pace with real-time changes, underscoring the need for more adaptive and proactive solutions.

Emerging technologies like AI and ML offer promising pathways to enable real-time adaptability. Frameworks like O-RAN [36] enable the capability and deployment of AI/ML through xApps on the NearRT-RIC and rApps on the NonRT- RIC, allowing for dynamic and efficient resource optimization. Additionally, initiatives such as the RAN Intelligence and Automation (RIA) [129] subgroup within the Telecom Infra Project (TIP) aim to foster an ecosystem that enhances RAN performance by leveraging AI/ML applications on O-RAN RIC frameworks. Similarly, the AI-RAN [28] community is working to create an AI-native RAN, focusing on optimizing RAN performance through AI applications (AI for RAN), integrating non-RAN AI workloads with RAN infrastructure (AI and RAN), and identifying key RAN requirements to deliver and benchmark AI applications over RAN connectivity and infrastructure (AI on RAN).

However, these approaches not only heavily depend on O-RAN architectures, requiring operators to navigate O-RAN-specific logic alongside 3GPP network standards, but also primarily focus on the RAN domain, overlooking the need for holistic optimization across the entire network, including CN and User Equipment (UE), as seen in use cases like end-to-end slice management. These limitations hinder the ability to achieve the real-time adaptability needed to meet evolving network demands. To address these challenges, the proposed AUTO-RAN enables real-time network adaptability by integrating a closed-loop autonomous mechanism within the O-RAN architecture while simplifying network optimization process for operators. By exposing only 3GPP logic, AUTO-RAN allows operators to optimize the network and service performance as they would in a traditional 3GPP system, without the need to handle O-RAN logic.

4.2.3 Seamless Integrating New Features and Use Cases

As new applications and technologies emerge, next-generation mobile networks are expected to evolve to meet their diverse and demanding requirements. For instance, eXtended Reality (XR) applications [23] in collaborative networks rely on ultra-low latency and high-bandwidth connections to provide seamless user interactions. To address these demands, 3GPP Release 18 introduced the concept of a PDU Set [18], which groups one or more Protocol Data Units (PDUs) carrying application-level information (e.g., video frames or slices for XR services). This approach enables the network to manage all PDUs within a Set cohesively while applying differentiated handling across PDU Sets, ensuring precise performance for XR applications.

In addition, technologies like digital twin, which creates virtual replicas of physical systems to simulate real-world environments (e.g., manufacturing, smart cities, and transport systems), are gaining prominence. Integrating digital twin technologies into mobile networks [107] requires highly reliable, low-latency, and synchronized data collection to enable faster-than-real-time simulation capabilities. 3GPP has studied scenarios involving digital twins [22], focusing on their integration with network management functions, identifying challenges, potential requirements, and possible solutions. Similarly, the O-RAN research group has published reports exploring use cases and enablers for digital twins in the RAN domain [76, 75].

Furthermore, the O-RAN architecture has facilitated the introduction of a range of innovative features such as control applications with capabilities like real-time control (e.g., dApp [81] and ChARM [54]) and embedded learning models (e.g., PandORA [142]), as well as FlexCtrl and IUP, discussed in Chapter 2 and 3. These advancements leverage O-RAN's flexibility to enhance the network optimization process. However, these advancements are introduced by either 3GPP, O-RAN, or both, with each ecosystem evolving independently and focusing on its own architecture for their implementation. This lack of coordination between the two ecosystems creates significant challenges for network operators, who have to manage separate systems to deploy new features and use cases.

The proposed AUTO-RAN simplifies this complexity by abstracting the O-RAN architecture while aligning with 3GPP standards. This design ensures that, regardless of whether new features or use cases originate from 3GPP or O-RAN, AUTO-RAN can seamlessly support them, as illustrated in Figure 4.1, thanks to its foundation on both 3GPP and O-RAN concepts. By abstracting complexities, AUTO-RAN simplifies the optimization process for network operators, enabling the natural adoption of new features and use cases. For example, FlexCtrl can be deployed within the O-RAN architecture without requiring network operators to determine where the control logic resides. Operators can optimize the performance by simply sending their policies to optimize performance based on 3GPP logic, or relying entirely on the automated mechanisms within AUTO-RAN without providing instructions.

4.3 Architecture

In this section, we present the high-level architecture of AUTO-RAN, as illustrated in Figure 4.3, with an example scenario where operator aims to optimize network and service performance. The proposed architecture extends and enhances the O-RAN architecture by introducing a novel autonomous control loop. Figure 4.3 outlines the message flows between network components within the proposed control loop and how operators can optionally interact with AUTO-RAN. Additionally, the figure highlights the responsibilities of vendors and operators in providing or managing the details of each network component.

4.3.1 Overview

The proposed AUTO-RAN introduces an autonomous control loop based on the O-RAN architecture, significantly simplifying network optimization tasks by abstracting the complexities of O-RAN. This enables network operators to focus on their 3GPP expertise without needing to understand detailed O-RAN specifications, as illustrated in Figure 4.3. Moreover, AUTO-RAN provides operators with the flexibility to control both legacy (without O-RAN) and integrated (with O-RAN) infrastructures seamlessly. In integrated infrastructures, it enables network operators to optionally interact with O-RAN components through 3GPP logic (represented by the dashed brown arrow in Figure 4.3) to observe network performance and enforce specific policies while leveraging advanced O-RAN functionalities.

The transformation between 3GPP and O-RAN network logic is achieved through the proposed autonomous control loop, which involves four key steps executed across network control applications (e.g., xApp, rApp) and underlying RAN nodes (e.g., E2-Node) through O-RAN specified interfaces (e.g., E2, A1). The messages transmitted between these network components carry content represented in either 3GPP or O-RAN logic. As illustrated in Figure 4.3, these four steps can be realized through four example control applications, and their details are outlined as follows:

Step 1: Awareness. The awareness xApp collects raw data (O-RAN logic) from E2-Nodes, processes it, and abstracts it into aggregated information from a 3GPP perspective. For example, latency data from individual packets (O-RAN logic) is translated into the average latency for a specific service (3GPP logic).

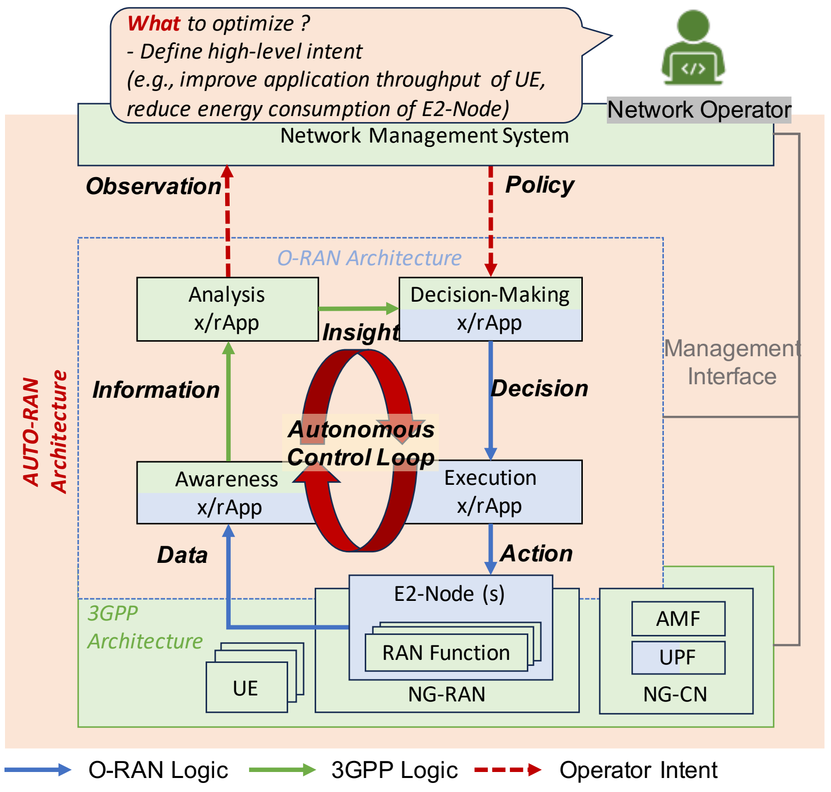


Figure 4.3: High-level architecture of AUTO-RAN for optimizing network and service performance scenarios.

Step 2: Analysis. This aggregated information is then passed to the analysis rApp, implemented entirely with 3GPP logic. The analysis rApp interprets the data and generates meaningful insights for the decision-making rApp. For example, it can calculate the percentage of packets exceeding a specific delay threshold (3GPP logic). Moreover, these insights can optionally be exposed to the Network Management System (NMS) , as shown in Step a of Figure 4.3, providing the operator with key observations about network performance for monitoring or further action.

Step 3: Decision-Making. The decision-making rApp determines the necessary actions to optimize the network based on the received insights and then passes these decisions to the execution xApp. For example, if the average delay for a specific service exceeds its delay budget (3GPP logic), the rApp generates a decision to address the issue (O-RAN logic) using methods such as predefined rules or ML models. Additionally, network operator can optionally influence the decision-making process by specifying policies (Step b in Figure 4.3) that guide the decisions made by the decision-making rApp.

Step 4: Execution. The execution xApp, implemented fully with O-RAN logic, executes the corresponding actions with required parameters to implement the control mechanism at the RAN function(s) within the E2-Node(s).

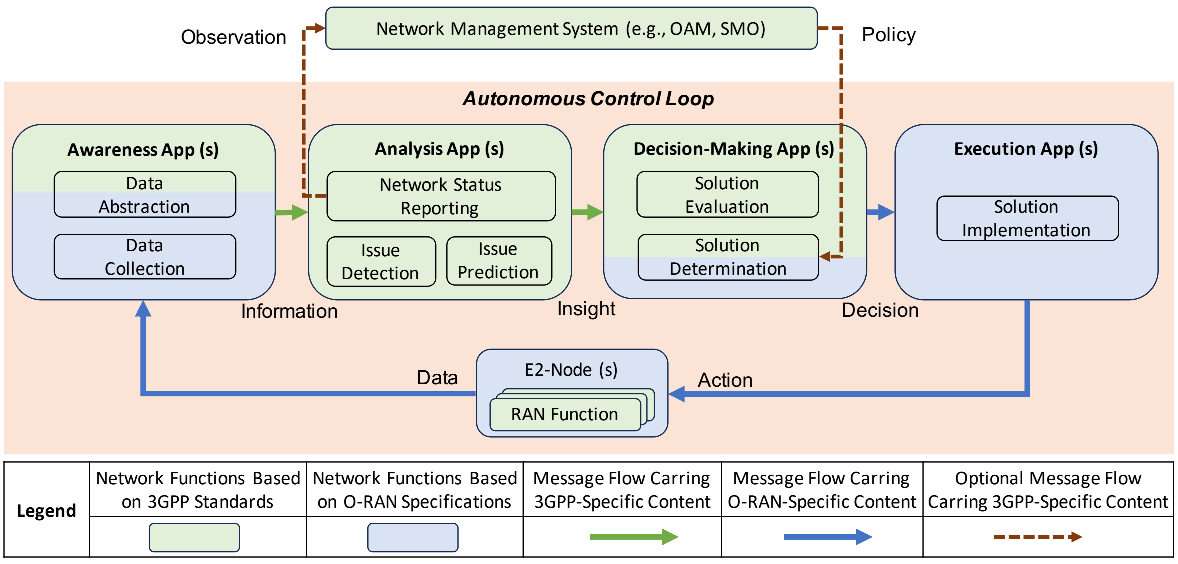


Figure 4.4: Details of autonomous control loop.

With its autonomous control loop, AUTO-RAN independently handles network optimization tasks, allowing O-RAN components to operate transparently within the integrated infrastructure, hidden from the operator's view. This seamless integration of the two ecosystems (3GPP and O-RAN) allows network operators to perform autonomous control optimizations while continuing to use familiar 3GPP-compliant management interfaces, effectively mitigating the complexities introduced by the O-RAN architecture.

4.3.2 Details of Autonomous Control Loop

Building on the overview of AUTO-RAN presented earlier, we now delve into the detailed functionalities of each step in the proposed autonomous control loop, which can be realized through control applications (referred to as Apps), as illustrated in Figure 4.4. These functionalities can be implemented within a single App or distributed across multiple Apps, depending on the preferences of the developer and vendor.

Data Collection and Abstraction. The first App - Awareness App - includes two primary functions: Data Collection and Data Abstraction. The Data Collection function gathers raw data from the E2-Node(s) via O-RAN APIs (e.g., E2-related APIs [40]), which is in compliance with O-RAN specifications. Meanwhile, the Data Abstraction function processes the collected data and transforms it into higher-level information that aligns with the 3GPP standards (e.g., 3GPP-defined 5G end-to-end key performance indicators [3]). These two functions can be further enhanced by incorporating an external database to store raw data and processed information, facilitating future analysis and extended use cases.

Issue Detection and Prediction. Based on the information provided by the first App, the second App — Analysis App — can detect current issues and predict potential future issues. The Issue Detection function focuses on past and/or present events, identifying what is or has happened. This function can be implemented using rule-based methods to detect specific conditions or patterns. The Issue Prediction function, on the other hand, focuses on future events, estimating what might happen next. This function typically requires AI/ML technologies to project future outcomes accurately. Both functions rely on historical and current information to perform their tasks effectively.

Network Status Reporting. Additionally, the Analysis App can report the network status via the Network Status Reporting function, which provides network insights (e.g., detected issues) to the next App for further evaluation and the development of solutions to resolve or mitigate these issues. It can also report the network status to the operator via the NMS, which seamlessly integrates O-RAN network information with the status of other components in the 3GPP network (e.g., UPF). Since the information is already translated into a 3GPP-compatible format (e.g., management interface), the operator can focus on processing 3GPP-related information without dealing with the complexities of the O-RAN architecture.

Solution Evaluation. Building on insights provided by the previous App, the third App — Decision-Making App — evaluates potential solutions and determines the most suitable one to address current or future issues. The Solution Evaluation function can leverage advanced technologies, such as digital twins, to simulate potential solutions in a virtual environment, assessing their effectiveness in resolving issues and optimizing network performance. Alternatively, it can use static methods, such as lookup tables, which record impacts of each solution on the network.

Solution Determination. Once the evaluation is complete, the Solution Determination function selects the optimal solution for the current network, ensuring alignment with predefined network policies and Service-Level Agreements (SLAs). This function plays a critical role in determining the final decision to be implemented within the O-RAN architecture by translating the solution from 3GPP logic (e.g., 3GPP-defined policy) into O-RAN logic (e.g., A1 policy). In other words, it takes the solution defined in 3GPP logic and identifies how to realize it within the O-RAN architecture, such as selecting the appropriate rApp/xApp to execute the control mechanism. Furthermore, operators can enforce additional policies on this function to influence the final decision.

Solution Implementation. Finally, the fourth App — Execution App — is responsible for processing the decisions provided by the previous App and and implementing the corresponding control actions to optimize network and service performance through the Solution Implementation function. This function specifies the detailed parameters required for control mechanisms, ensuring alignment with O-RAN APIs. Additionally, it manages conflicts in the underlying network by recording previously applied actions to prevent potential conflicts and adjusting parameters when necessary.

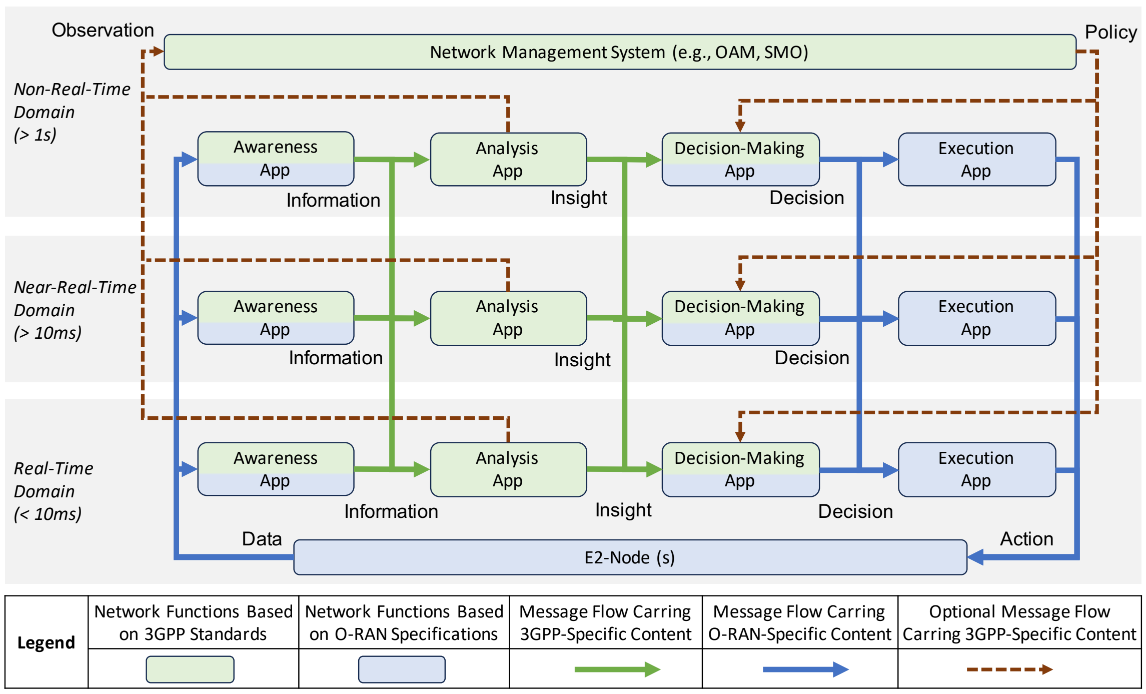

Observation
Network Management System (e.g., OAM, SMO)
Policy
Non-Real-Time Domain (> 1s)
Near-Real-Time Domain (> 10ms)
Real-Time Domain (< 10ms)
Awareness App
Analysis App
Decision-Making App
Execution App
Information
Insight
Decision
Data
E2-Node (s)
Action
Legend
Network Functions Based on 3GPP Standards
Network Functions Based on O-RAN Specifications
Message Flow Carring 3GPP-Specific Content
Message Flow Carring O-RAN-Specific Content
Optional Message Flow Carring 3GPP-Specific Content

Figure 4.5: Deployment options of autonomous control loop.

4.3.3 Deployment Flexibility

The proposed autonomous control loop not only abstracts the complexities introduced by O-RAN architecture, but also provides the deployment flexibility, allowing the Apps to be deployed in various domains depending on the required control latency (following the concept of FlexCtrl introduced in Section 2.5), as shown in Figure 4.5. For latency requirements in the non-real-time domain (>1 s), Apps can be implemented as rApps on the NonRT-RIC or as functions within OAM/SMO. For near-real-time latency requirements (>10 ms), Apps can be deployed as xApps on the NearRT-RIC. Finally, for real-time latency requirements (<10 ms), Apps can be realized as internal functions, such as dApps [81], on the E2-Node(s).

For example, the autonomous control loop illustrated in Figure 4.3 is implemented using two xApps on the NearRT-RIC and two rApps on the NonRT-RIC. To achieve real-time network adaptability, operators can deploy all four steps/Apps in the real-time domain, or network vendors can implement them directly within the E2-Node(s). While this approach enhances adaptability, it may increase the computational load on the E2-Node(s). Alternatively, real-time network adaptability can be enabled by moving more complex Apps, such as analysis and decision-making Apps capable of handling 3GPP-specific content, to the near- or non-real-time domains. This allows the execution App to remain close to the E2-Nodes, enabling efficient real-time action while offloading intensive processing to higher domains.

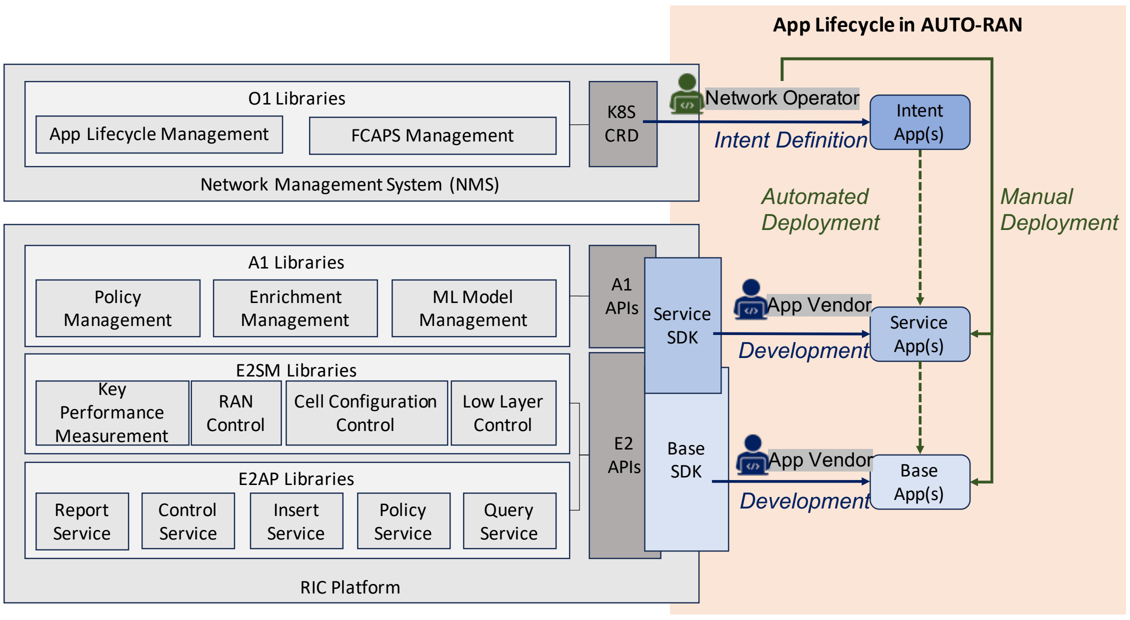


Figure 4.6: Control application lifecycle in AUTO-RAN: integration with the O-RAN ecosystem and the roles of network vendors and operators.

4.3.4 Control Application Lifecycle

Control applications, referred to as Apps, play a crucial role in establishing the proposed autonomous control loop, which consists of four key functionalities: awareness, analysis, decision-making, and execution, as discussed in Section 4.3.1. To implement these functionalities efficiently, AUTO-RAN introduces a hierarchical control application lifecycle with three levels of abstraction: Base Apps, Service Apps, and Intent Apps. As illustrated in Figure 4.6 and detailed in Table 4.3, this progression simplifies operations for network operators by gradually abstracting O-RAN logic — from low-level control in Base Apps to higher-level abstraction in Service Apps and full automation in Intent Apps. By adopting this structured approach, AUTO-RAN enables fully automated and intent-driven control, ensuring scalability and adaptability in dynamic network environments.

4.3.4.1 Base App: Low-Level Development with Full Control

Base Apps are developed using the Base SDK, which is built on top of E2 APIs. These E2 APIs include standardized E2AP procedures, such as RIC subscription and RIC control, and E2SM data structures, including KPM and RC. Since Base Apps are tightly coupled with these standardized procedures, developers, such as App vendors, must have strong expertise in E2AP and E2SM to properly utilize the Base SDK. To develop Base Apps, developers need to generate subscription data following KPM SM specifications to perform report services and collect specific measurement data, while also constructing control messages based on RC SM specifications to execute control services, such as triggering handover operations in underlying E2-Nodes.

Table 4.3: Characteristics of control application in AUTO-RAN.

Attributes	**Base App**	**Service App**	**Intent App**
Provided By	App Vendor	App Vendor	Network Operator
Built On	Base SDK	Service SDK	Kubernetes CRD
Operator Interaction	Imperative	Imperative	Declarative
Purpose	E2-related functions	Service-based & A1-related functions	High-level intents
Examples	Data collection and control execution	Data analysis and control decision-making	Policy enforcement
Lines of Code	150+	40+	15+
O-RAN/RIC Expertise	Proficient	Basic	None
3GPP/NMS Expertise	None	Basic	Proficient
Configuration Method	Manual	Manual	Automatic
Deployment Model	Static	Static	Dynamic

Although Base Apps offer fine-grained control and ensure compliance with standardized data collection and control execution methods, they require significant development effort, often involving more than 100 lines of code for a single control procedure. From the perspective of network operators, Base Apps are not easily scalable since each implementation typically supports only a single scenario, such as collecting throughput metrics from one specific E2-Node. If network operators need to scale data collection across multiple E2-Nodes or adjust the collected metrics, they must request App vendors to modify the code accordingly and find a way to automate deployment instead of relying on manual intervention. This limitation makes Base Apps less flexible for large-scale deployments where network conditions frequently change.

4.3.4.2 Service App: Simplified Development with High-Level Abstraction

Service Apps are developed using the Service SDK, which builds on the Base SDK and introduces A1 APIs. The Service SDK provides a higher-level abstraction of the Base SDK, allowing developers to focus on the development of service-based and A1-related functions without dealing with E2-level complexities. Unlike Base Apps, which require deep technical knowledge of E2AP and E2SM, Service Apps prioritize usability by shifting the focus from *how to write* low-level procedures to *how to use* pre-built functions efficiently. Developers only need basic knowledge of RIC platforms, as the Service SDK encapsulates underlying E2 functionalities, enabling them to concentrate on advanced tasks such as data processing, traffic analysis, and control decision-making.

Service Apps also allow network operators to define A1 policies, which are deployed manually through the Network Management System (NMS). Once deployed, a Service App processes the defined policies and executes the corresponding actions based on its built-in service-based functions. For example, if a network operator provides a slice policy, the Service App can dynamically manage slice control operations. However, despite this abstraction, network operators must still have knowledge of the A1 interface, as they are responsible for manually configuring policies and interacting with the O-RAN ecosystem. While Service Apps significantly reduce the development effort required compared to Base Apps, they still require some level of operator involvement for policy configuration and execution.

4.3.4.3 Intent App: Fully Automated, Code-Free Deployment

Intent Apps are defined using Kubernetes Custom Resource Definitions (CRDs), allowing network operators to express high-level intent through a YAML file instead of writing code. This further abstracts the complexities of both Base and Service Apps, enabling network operators to interact with the O-RAN ecosystem declaratively, without requiring any knowledge of RIC platforms. By leveraging cloud-native technologies, Intent Apps automate network configuration and deployment, enable policy-driven service orchestration, and dynamically deploy applications based on network intent.

Once an Intent App is defined and deployed, it automatically triggers the necessary Service App based on the specified intent. The Service App then translates this intent into an A1 policy and executes the corresponding actions. If required, the Service App can dynamically deploy a Base App to execute specific low-level control actions. For example, if an Intent App defines an intent to collect resource usage metrics from a specific gNB, the deployed Service App interprets the policy, selects the appropriate Base App, and dynamically programs it to carry out the data collection process.

4.4 Proof-of-Concept Implementation

This section elaborates on the implementation of AUTO-RAN, integrating the proposed autonomous control loop within the O-RAN architecture to automate network optimization process.

4.4.1 Platforms and Validation

The proposed autonomous control loop is built on FlexRIC [134], an open-source platform that provides O-RAN-compliant interfaces (e.g., E2) and frameworks for monitoring and controlling RAN nodes (referred to as E2-Nodes in O-RAN terminology). The choice of RAN and CN technologies depends on the use case. For the RAN slicing use case, we leverage OpenAirInterface (OAI) [114], an open-source platform that provides 5G end-to-end systems compliant with 3GPP standards. OAI also integrates O-RAN compliant interfaces, enabling seamless communication with O-RAN components such as xApps on the NearRT-RIC. For the mobility management use case, we utilize the Amarisoft Callbox [51], a commercial 5G system also compliant with 3GPP standards. However, Amarisoft does not natively support O-RAN interfaces, instead providing a WebSocket-based network API for operator control. To bridge this gap, we implemented a proxy agent based on FlexRIC, allowing the Amarisoft RAN to be controlled in compliance with O-RAN specifications. Additionally, AUTO-RAN is validated using Commercial Off-The-Shelf (COTS) UEs, ensuring its applicability in real-world scenarios.

4.4.2 Implementation Details

The detailed implementation of autonomous control loop functionalities in AUTO-RAN is outlined below, covering two use cases: RAN slicing and mobility management. RAN slicing operates without network operator interaction, while mobility management involves operator input, demonstrating that AUTO-RAN can both autonomously optimize the network and incorporate operator-defined policies when needed.

4.4.2.1 Use case: Automated RAN Slicing

As illustrated in Figure 4.7, a proof-of-concept for AUTO-RAN is implemented by integrating the autonomous control loop within a single xApp on the NearRT-RIC. To avoid additional latency caused by message exchanges across the network between multiple xApps and rApps, the four key steps — Awareness, Analysis, Decision-Making, and Execution — are consolidated within a single xApp. Although these functionalities could be embedded directly within the E2-Node, as detailed in Section 4.3.3, implementing the autonomous control loop at the xApp level enables effective control of scenarios involving multiple E2-Nodes.

Multi-Threading. The implemented autonomous control loop, as illustrated in Figure 4.7, demonstrates its applicability through the use case of automatically enabling network slicing to optimize network and service performance. An xApp was developed with two threads to handle monitor and control tasks simultaneously. A thread-safe queue facilitates communication between these threads, storing information such as traffic types and radio resource utilization for further analysis. This design allows the control thread to analyze data, make decisions, and send the necessary control messages to the E2-Nodes without waiting for the monitor thread to complete its tasks. Meanwhile, the monitor thread continuously collects underlying data in parallel.

Monitor Thread. The monitor thread includes the first step (Awareness) and its associated functions, which are capable of processing O-RAN specific content (e.g., E2AP and E2SM). These functions subscribe to RAN functions via corresponding SMs, following the RIC report service procedure, and periodically receive RIC indication messages containing raw data from E2-Nodes. Upon receiving RIC indication messages, these functions decapsulates the raw data and aggregates it into 3GPP-specific information. This includes metrics such as per-second radio resource utilization for each UE and the types of traffic received

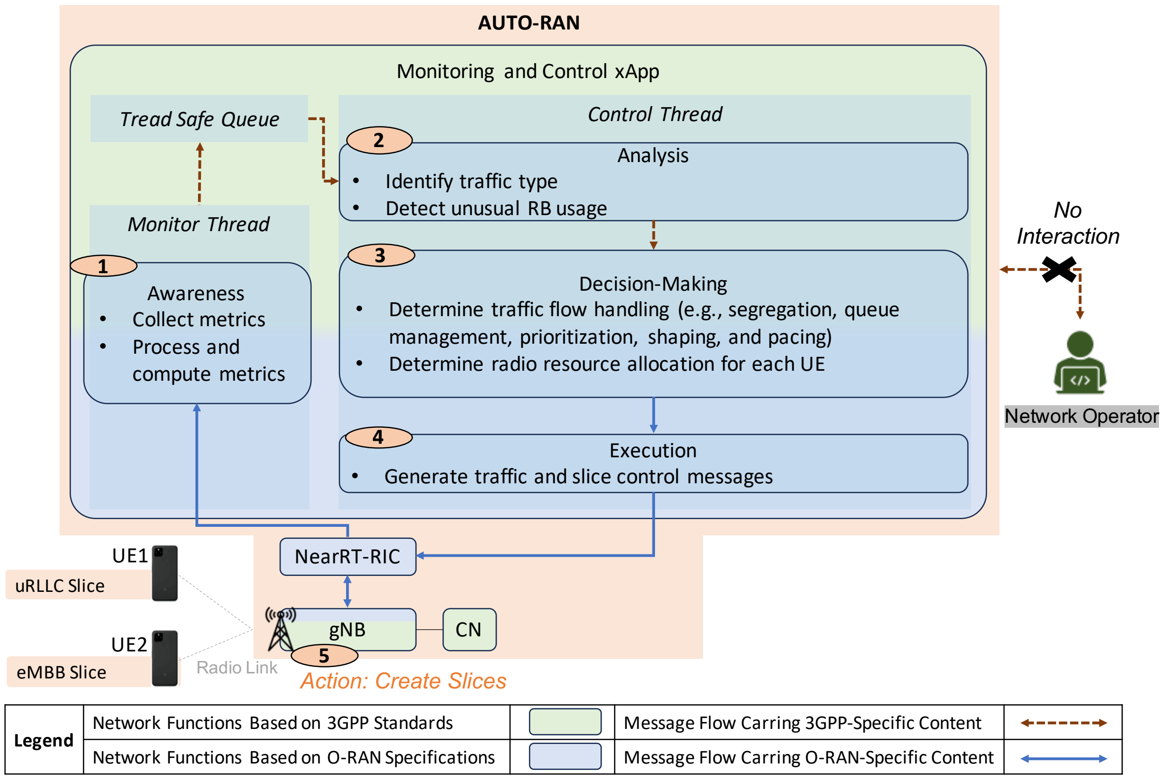
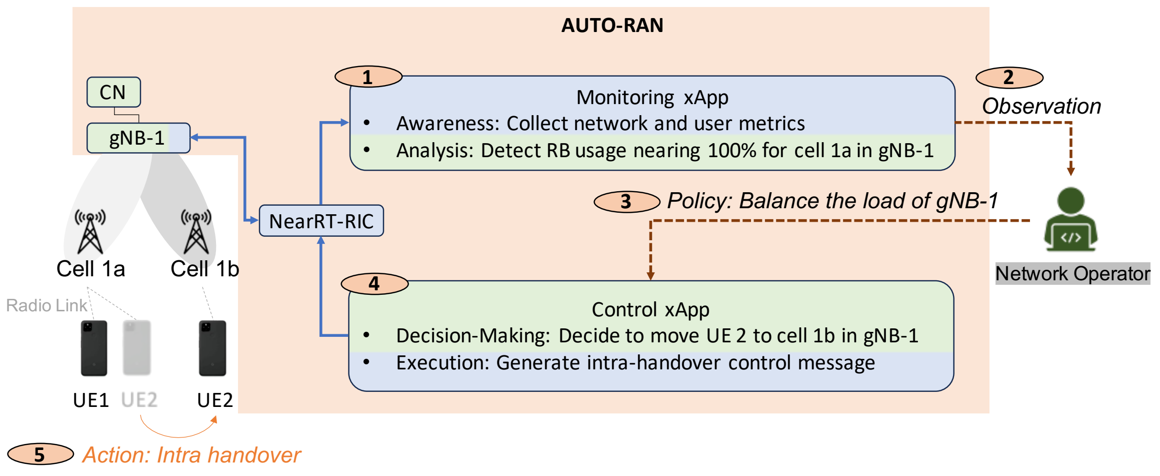

AUTO-RAN
Monitoring and Control xApp
Tread Safe Queue
Control Thread
2
Analysis
• Identify traffic type
• Detect unusual RB usage
Monitor Thread
1
Awareness
• Collect metrics
• Process and compute metrics
3
Decision-Making
• Determine traffic flow handling (e.g., segregation, queue management, prioritization, shaping, and pacing)
• Determine radio resource allocation for each UE
No Interaction
Network Operator
4
Execution
• Generate traffic and slice control messages
UE1
uRLLC Slice
UE2
eMBB Slice
NearRT-RIC
gNB
CN
5
Radio Link
Action: Create Slices
Legend
Network Functions Based on 3GPP Standards
Network Functions Based on O-RAN Specifications
Message Flow Carring 3GPP-Specific Content
Message Flow Carring O-RAN-Specific Content

Figure 4.7: Implementation details of AUTO-RAN in RAN slicing use case.

during that time. The abstracted information is then pushed to the thread-safe queue, where it can be queried by the control thread for further processing.

Control Thread. The control thread includes the rest three steps of the autonomous control loop: Analysis, Decision-Making, and Execution. In the second step (Analysis), the functions retrieve data from the thread-safe queue, identify issues such as new traffic types or SLA violations, and generate network insights for the next step, including identifying active services and their required performance. For example, the system may detect that UE 1 is using a uRLLC service (e.g., online gaming), while UE 2 is using an eMBB service (e.g., video streaming). During the third step (Decision-Making), the functions determine final control decisions based on predefined rules to address issues, such as handling the arrival of a new service through slicing policy control. These decisions, grounded in O-RAN specific logic, include selected control options for traffic management and resource allocation. In the fourth step (Execution), the functions generate control actions based on the given decisions, encapsulate them into RIC control messages, and execute the RIC control service procedure with the underlying RAN functions, following O-RAN specifications. Finally, the gNB carries out the control actions, creating two slices: UE 1 is assigned to the slice for uRLLC service, and UE 2 is assigned to the slice for eMBB service.


AUTO-RAN
CN
gNB-1
Cell 1a
Cell 1b
Radio Link
UE1
UE2
UE2
NearRT-RIC
1
Monitoring xApp
• Awareness: Collect network and user metrics
• Analysis: Detect RB usage nearing 100% for cell 1a in gNB-1
2
Observation
3
Policy: Balance the load of gNB-1
Network Operator
4
Control xApp
• Decision-Making: Decide to move UE 2 to cell 1b in gNB-1
• Execution: Generate intra-handover control message
5
Action: Intra handover

Legend				
Legend	Network Functions Based on 3GPP Standards	🟩	Message Flow Carring 3GPP-Specific Content	⇠⇢
	Network Functions Based on O-RAN Specifications	🟦	Message Flow Carring O-RAN-Specific Content	⟷

Figure 4.8: Implementation details of AUTO-RAN for mobility management in a load balancing scenario.

4.4.2.2 Use Case: Automated Mobility Management

In this use case, AUTO-RAN is implemented by integrating the autonomous control loop within two xApps on the NearRT-RIC, as illustrated in Figure 4.9 and Figure 4.8. The first xApp monitors network and user performance, handling awareness and analysis tasks, while the second xApp manages handover control, covering decision-making and execution tasks. To demonstrate mobility management use cases through handover control, two scenarios are implemented: intra-handover control for load balancing and inter-handover control for energy saving. Moreover, both scenarios allow seamless interaction with the network operator.

Load Balancing. Figure 4.8 illustrates five steps in the load balancing scenario. In Step 1, the monitoring xApp collects network and user metrics, including Radio Resource Block (RB) usage, the bitrate of connected UEs, and (neighbor) cell information. It then processes this data to identify potential issues. In Step 2, if a network issue is detected, such as RB usage nearing 100% for Cell 1a in gNB-1, the monitoring xApp reports it to the network operator. In Step 3, the network operator responds by sending a policy directive to AUTO-RAN, instructing it to balance the load on gNB-1. In Step 4, the control xApp processes the policy and decides to initiate handover control, moving UE 2 from Cell 1a to Cell 1b in gNB-1 to redistribute the load. It then generates the required control message following O-RAN RC SM specifications and sends it to gNB-1. In Step 5, the gNB-1 receives the control action and executes the intra-handover, seamlessly transferring UE 2 from Cell 1a to Cell 1b without service interruption. Unlike conventional handover mechanisms, where the UE triggers the handover, this approach enables the network operator to proactively optimize network performance based on real-time network conditions.

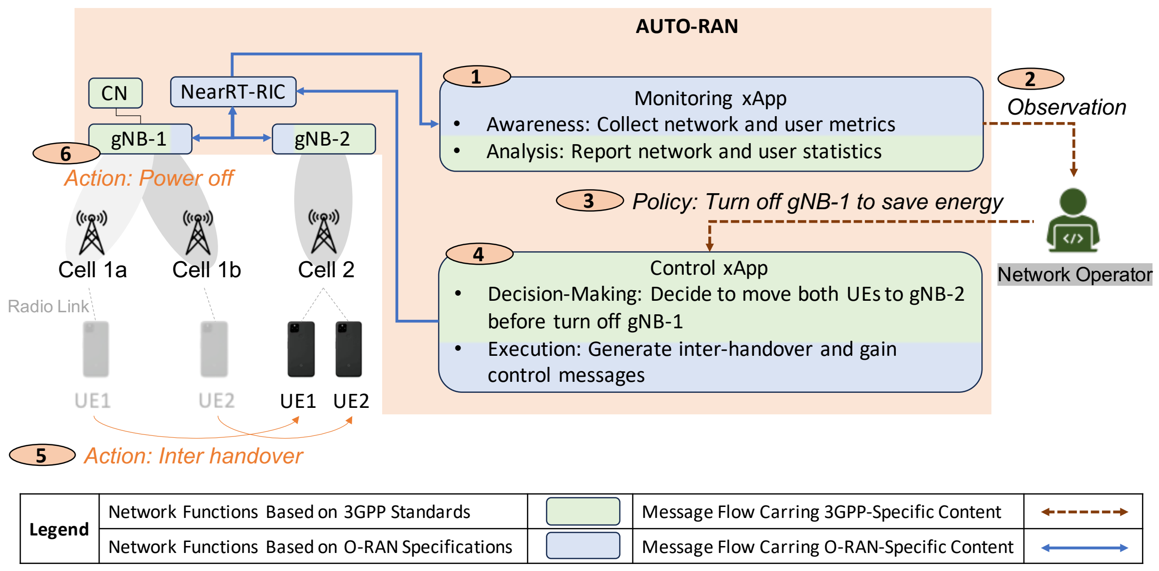

AUTO-RAN
CN
NearRT-RIC
gNB-1
gNB-2
6
Action: Power off
Cell 1a
Cell 1b
Cell 2
Radio Link
UE1
UE2
UE1
UE2
5
Action: Inter handover
1
Monitoring xApp
• Awareness: Collect network and user metrics
• Analysis: Report network and user statistics
2
Observation
3
Policy: Turn off gNB-1 to save energy
Network Operator
4
Control xApp
• Decision-Making: Decide to move both UEs to gNB-2 before turn off gNB-1
• Execution: Generate inter-handover and gain control messages
Legend
Network Functions Based on 3GPP Standards
Network Functions Based on O-RAN Specifications
Message Flow Carring 3GPP-Specific Content
Message Flow Carring O-RAN-Specific Content

Figure 4.9: Implementation details of AUTO-RAN for mobility management in an energy-saving scenario.

Energy Saving. The energy-saving scenario is implemented in five steps, as shown in Figure 4.9. In Step 1, the monitoring xApp collects network metrics, similar to the load balancing scenario, and processes this data into higher-level insights before reporting to the network operator. In Step 2, the network operator observes a low-traffic period (e.g., nighttime), during which a high-powered gNB is not needed to support extensive coverage and multiple UEs. In Step 3, recognizing the low UE density, the network operator sends a policy directive to turn off gNB-1 to reduce energy consumption. In Step 4, the control xApp processes the policy and initiates handover control, transferring all UEs from gNB-1 to gNB-2 before shutting down gNB-1. It then generates the required control messages and sends them to gNB-1. In Step 5, the gNB-1 executes the handover procedure (Xn handover), seamlessly transferring all UEs from gNB-1 to gNB-2 without service disruption. Afterward, it reduces the transmission power of gNB-1 to zero. Compared to the traditional approach of directly turning off gNB-1, which may cause UEs to lose service and reconnect independently, this method ensures seamless service continuity while enabling the network operator to optimize performance and reduce energy consumption efficiently.

4.4.3 Reduced Control Application Development Complexity: Examples of xApp and rApp

In this section, we present example code for control applications, as discussed in Section 4.3.4, including Base, Service, and Intent Apps. We demonstrate two scenarios: one for data monitoring and another for slice control. For each scenario, we showcase three Apps to highlight their differences and illustrate how they progressively abstract the underlying logic. The Base and Service Apps, functioning as xApps, are developed based on

FlexRIC[134], while the Intent Apps, acting as an rApps, are built on MAESTRO [63][2].

4.4.3.1 Data Monitoring Apps

To collect the required metrics, we follow the standardized RIC report service procedure for KPM SM and develop the corresponding Apps, as detailed below.

Base App (xApp). Developing a Base App requires four key steps, as shown in Listing 4.1:

1. Start: Initialize the xApp and retrieve information about connected E2-Nodes.
2. RIC Report Service: Prepare the necessary details, such as the event trigger and action definition, for the KPM SM subscription request message, which specifies the required measurement data.
3. Data Collection: Process the received indication messages within the callback function, which was defined during the subscription request.
4. Stop: Send a subscription delete request to terminate data collection from E2-Nodes and stop the xApp.

For simplicity, the code related to the callback function is not included in Listing 4.1. Further details can be found in the FlexRIC repository[3].

Service App (xApp). Developing a Service App involves three key steps. However, compared to the Base App, it simplifies much of O-RAN's complexity, allowing developers to avoid dealing with intricate data structures and procedures. As shown in Listing 4.2, the required steps are as follows:

1. Start: Initialize the xApp, specify the current use case, and retrieve information about connected E2-Nodes.
2. Data Collection: Obtain measurement data directly by calling a function provided by the Service SDK and specifying the required measurement name.
3. Stop: Terminate the xApp.

Intent App (rApp). Unlike the Base and Service Apps, an Intent App[4] allows operators to define measurement data without coding by using a YAML format, as shown in Listing 4.3.

[2]A cloud-native platform that includes SMO and provides an intent-based framework leveraging Large Language Models (LLMs) for network automation.

[3]https://gitlab.eurecom.fr/mosaic5g/flexric/-/blob/br-flexric/examples/xApp/c/monitor/

[4]Demo video available at:https://youtu.be/3oHtkL39eo0

```c

/*********************************/
//* 0. Define parameters        *//
/*********************************/
uint64_t const const e2node_nbid = 50;
uint64_t const report_period_time_ms = 1000;
char* const measurement_name[3] = {
    "DRB.PdcpSduVolumeDL",
    "DRB.UEThpDl",
    "RRU.PrbTotDl"};

int main(int argc, char *argv[])
{
  /*********************************/
  //* 1. Start the xApp           *//
  /*********************************/
  // 1a. Initialize xApp
  fr_args_t args = init_fr_args(argc, argv);
  init_xapp_api(&args);
  // 1b. Get connected E2-Nodes infroamtion
  e2_node_arr_xapp_t nodes = e2_nodes_xapp_api();

  /*********************************/
  //* 2. Start RIC Report Service *//
  /*********************************/
  // 2a. Define a handler to store the RIC Request ID
  sm_ans_xapp_t* oran_sm_handle = calloc(nodes.len, sizeof(sm_ans_xapp_t));
  // 2b. Subscribe KPM SM on specified E2-Node
  for (int i = 0; i < nodes.len; i++) {
    if (nodes.n[i].id.nb_id.nb_id == e2node_nbid) {
        // 2d. Generate KPM subscription data: event trigger
        kpm_sub_data_t kpm_sub = {0};
        kpm_sub.ev_trg_def = gen_kpm_ev_trig(report_period_time_ms);
        // 2e. Generate KPM subscription data: action definition
        kpm_sub.sz_ad = 1;
        kpm_sub.ad = calloc(1, sizeof(kpm_act_def_t));
        *kpm_sub.ad = gen_kpm_act_def(measurement_name,
                                      report_period_time_ms,
                                      FORMAT_4_ACTION_DEFINITION);

        // 2d. Send KPM subscription request, and indicate the callback function to get the return indicaiton message
        oran_sm_handle[i] = report_sm_xapp_api(&nodes.n[i].id,
                                               SM_KPM_ID,
                                               &kpm_sub,
                                               sm_cb_kpm);
   }
  }

  /*********************************/
  //* 3. Get the measurement data from the callback function sm_cb_kpm() *//
  /*********************************/

  /*********************************/
  //* 4. Exit                     *//
  /*********************************/
  // 4a. Send subcription delete request
  for(int i = 0; i < nodes.len; ++i)
    rm_report_sm_xapp_api(oran_sm_handle[i].u.handle);
  free(oran_sm_handle);
  // 4b. Stop the xApp
  while(try_stop_xapp_api() == false)
    usleep(1000);
}
```

Listing 4.1: Base xApp Example (C) for monitoring required measurement data.

```c

/*********************************/
//* 0. Define parameters        *//
/*********************************/
uint64_t const const e2node_nbid = 50;

int main(int argc, char *argv[])
{
  /*********************************/
  //* 1. Start the xApp           *//
  /*********************************/
  // 1a. Initialize xApp
  init_xapp_sdk(argc, argv);
  // 1b. Specify the current use case and get connected E2-Nodes information
  arr_node_data_t arr = node_data_xapp_sdk(MONITOR_USE_CASE_e);

  /*********************************/
  //* 2. Get the measurement data on specified E2-Node *//
  /*********************************/
  for(int i = 0; i < arr.sz; ++i) {
    global_e2_node_id_sdk_t const* node = &arr.n[i].node;

    if (arr.n[i].node.nb_id.nb_id == e2node_nbid) {
        // 2a. Get the measurement data related to E2-Node
        float pdcp_sdu_vol_dl = e2_node_mntr_xapp_sdk(node, PDCP_SDU_VOLUME_DL);
        // 2b. Get the measurement data related to UEs
        for(int j = 0; j < arr.n[i].sz_ue; ++j) {
            ue_id_e2sm_sdk_t const* ue = &arr.n[i].ue[j].ue;

            float thp_dl = ue_mntr_xapp_sdk(node, ue, UE_THP_DL);
            int prb_tot_dl = ue_mntr_xapp_sdk(node, ue, PRB_TOT_DL);
        }
    }
  }

  /*********************************/
  //* 3. Exit                     *//
  /*********************************/
  free_arr_node_data (&arr);
  return EXIT_SUCCESS;
}
```

Listing 4.2: Service xApp Example (C) for monitoring required measurement data.

```yaml
apiVersion: ric.trirematics.io/v1
kind: MonitoringJob
metadata:
    name: intent
    namespace: trirematics
spec:
    tasks:
        -   networksMetricsMap:
                gnb1.regionname:
                    - Physical Resource Utilization
                    - Data Volume
                gnb2.regionname:
                    - Physical Resource Utilization
                    - Data Volume
                    - Throughput
```

Listing 4.3: Intent rApp Example (YAML) for specifying required measurement data.

4.4.3.2 Slice Control Apps

To manage slices within the RAN, we follow the standardized RIC control service procedure for RC SM and develop the corresponding Apps, as detailed below.

Base App (xApp). Similar to the monitoring case, developing a Base App for slice control requires three key steps, as shown in Listing 4.4:

1. Start: Initialize the xApp and retrieve information about connected E2-Nodes.
2. RIC Control Service: Prepare the necessary details, such as the control header and message, for the RC SM control request, which specifies the slice configurations.
3. Stop: Terminate the xApp.

Service App (xApp). Similar to the monitoring case, developing a Service App also involves three key steps, as shown in Listing 4.5:

1. Start: Initialize the xApp, specify the current use case, and retrieve information about connected E2-Nodes.
2. Control Execution: Perform slice control by calling a function provided by the Service SDK and specifying the slice configurations.
3. Stop: Terminate the xApp.

Moreover, the Service App can be further extended to expose interactive APIs, enabling operators to use it[5] without needing to write code from scratch.

Intent App (rApp). Unlike the Base and Service Apps, an Intent App allows operators to define slice policies using a YAML format, as shown in Listing 4.6.

[5]Demo video available at:https://youtu.be/kbh31hSxVFI

```c

/*********************************/
//* 0. Define parameters        *//
/*********************************/
uint64_t const const e2node_nbid = 50;
uint32_t const control_service_style = 2; // Radio resource allocation control
uint16_t const control_action_iud = 6; // Slice level PRB quota
typedef struct {
    char* sst;
    char* sd;
    int max_ratio;
    int min_ratio;
    int dedicated_ratio;
} rrm_policy_ratio_t;
rrm_policy_ratio_t const nssai_configs[3] = {
    {"1", "0", 20, 20, 20},
    {"1", "1", 40, 20, 0},
    {"2", "1", 100, 0, 0}
};

int main(int argc, char *argv[])
{
  /*********************************/
  //* 1. Start the xApp           *//
  /*********************************/
  // 1a. Initialize xApp
  fr_args_t args = init_fr_args(argc, argv);
  init_xapp_api(&args);
  // 1b. Get connected E2-Nodes infroamtion
  e2_node_arr_xapp_t nodes = e2_nodes_xapp_api();

  /*********************************/
  //* 2. Start RIC Control Service *//
  /*********************************/
  // 2a. Generate RC control header
  rc_ctrl_req_data_t rc_ctrl = {0};
  rc_ctrl.hdr = gen_rc_ctrl_hdr(control_service_style,
                                control_action_iud, FORMAT_1_E2SM_RC_CTRL_HDR);
  // 2b. Generate RC control message
  rc_ctrl.msg = gen_rc_ctrl_msg(nssai_configs,
                                FORMAT_1_E2SM_RC_CTRL_MSG);

  // 2c. Send control request to specific E2-Node
  for (int i = 0; i < nodes.len; i++) {
    if (nodes.n[i].id.nb_id.nb_id == e2node_nbid) {
      control_sm_xapp_api(&nodes.n[i].id,
                          SM_RC_ID,
                          &rc_ctrl);
    }
  }

  /*********************************/
  //* 3. Exit                     *//
  /*********************************/
  // 3a. Stop the xApp
  while(try_stop_xapp_api() == false)
    usleep(1000);
}
```

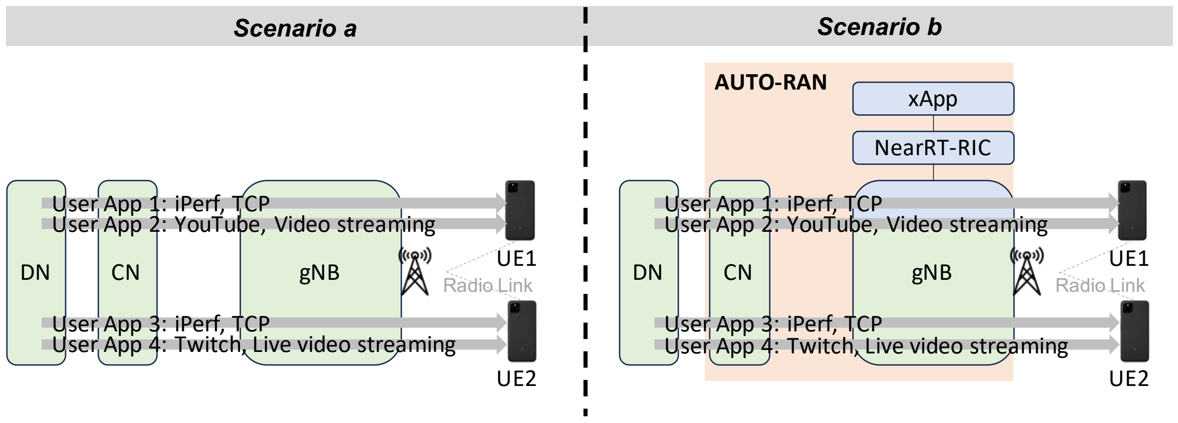
Listing 4.4: Base xApp Example (C) for slice control.

```c

/*********************************/
//* 0. Define parameters        *//
/*********************************/
uint64_t const const e2node_nbid = 50;
typedef struct {
    char* sst;
    char* sd;
    int max_ratio;
    int min_ratio;
    int dedicated_ratio;
} rrm_policy_ratio_t;
rrm_policy_ratio_t const nssai_configs[3] = {
    {"1", "0", 20, 20, 20},
    {"1", "1", 40, 20, 0},
    {"2", "1", 100, 0, 0}
};

int main(int argc, char *argv[])
{
  /*********************************/
  //* 1. Start the xApp           *//
  /*********************************/
  // 1a. Initialize xApp
  init_xapp_sdk(argc, argv);
  // 1b. Specify the current use case and get connected E2-Nodes information
  arr_node_data_t arr = node_data_xapp_sdk(SLICE_USE_CASE_e);

  /*********************************/
  //* 2. Send control request to specific E2-Node *//
  /*********************************/
  for (int i = 0; i < arr->sz; i++) {
    if (arr.n[i].node.nb_id.nb_id == e2node_nbid) {
      slice_xapp_sdk(&arr.n[i].node, nssai_configs);
    }
  }

  /*********************************/
  //* 3. Exit                     *//
  /*********************************/
  free_arr_node_data(&arr);
  return EXIT_SUCCESS;
}
```

Listing 4.5: Service xApp Example (C) for slice control.

```yaml
apiVersion: ric.trirematics.io/v1
kind: PolicyJob
metadata:
    name: intent
    namespace: trirematics
spec:
    tasks:
        - policyStatements:
            gnb1.regionname:
                - sliceid:
                    - nssaiId:
                        - sst: "1"
                        - sd: "000000"
                    - maxDlThptPerUe: 35
                - sliceid:
                    - nssaiId:
                        - sst: "1"
                        - sd: "000001"
                    - maxDlThptPerUe: 65
                - sliceid:
                    - nssaiId:
                        - sst: "2"
                        - sd: "000001"
                    - maxDlThptPerUe: 160
```

Listing 4.6: Intent rApp Example (YAML) for specifying slice policy.



Components	CN, gNB	UE1, UE2	NearRT-RIC, xApps
Platform/Hardware Used	OpenAirInterface	Pixel 5	FlexRIC

Figure 4.10: Experiment setup for slicing scenarios with and without AUTO-RAN.

4.5 Experimental Evaluation

In this section, we present preliminary results from our over-the-air testbed, demonstrating how AUTO-RAN enables automated control mechanisms for two use cases: RAN slicing[6] and mobility management[7]. These are achieved through xApps that support real-time adaptability and abstract O-RAN complexities, enabling declarative optimization process.

4.5.1 Enabled Real-Time Traffic Detection and RAN Slicing

Setup and Workload. We demonstrate the real-time network adaptability of AUTO-RAN by developing an xApp (Figure 4.7) that implements the proposed autonomous control loop mechanism[8]. The xApp supports real-time traffic detection and dynamic enforcement of slicing policies, enabling fully automated slice and traffic control within the RAN. To evaluate its effectiveness in a practical scenario, we set up two experimental configurations, illustrated in Figure 4.10, comparing network and user performance with AUTO-RAN xApp (Scenario b) and without it (Scenario a). In Scenario a, two UEs connect to the gNB, each running two user applications with downlink traffic from the Data Network (DN). UE 1 runs an iPerf TCP flow and the YouTube application streaming video, while UE 2 runs the same iPerf flow and the Twitch application streaming live content. In Scenario b, the AUTO-RAN xApp is deployed to autonomously detect traffic types, make control decisions, and dispatch corresponding slice and traffic policies (Table 4.4) to the gNB. These policies enable real-time traffic segregation and dynamic RAN slicing with appropriate radio resource allocation, tailored to application-level requirements.

[6]Demo video available at: https://youtu.be/iwwC9Tw21OM
[7]Demo video available at: https://youtu.be/hlLt--WSQPc
[8]We utilize the standardized KPM SM, along with two customized SMs from FlexRIC: Traffic Control (TC) and Slice Control (SC).

Table 4.4: Slice and traffic policies applied by the xApp in AUTO-RAN.

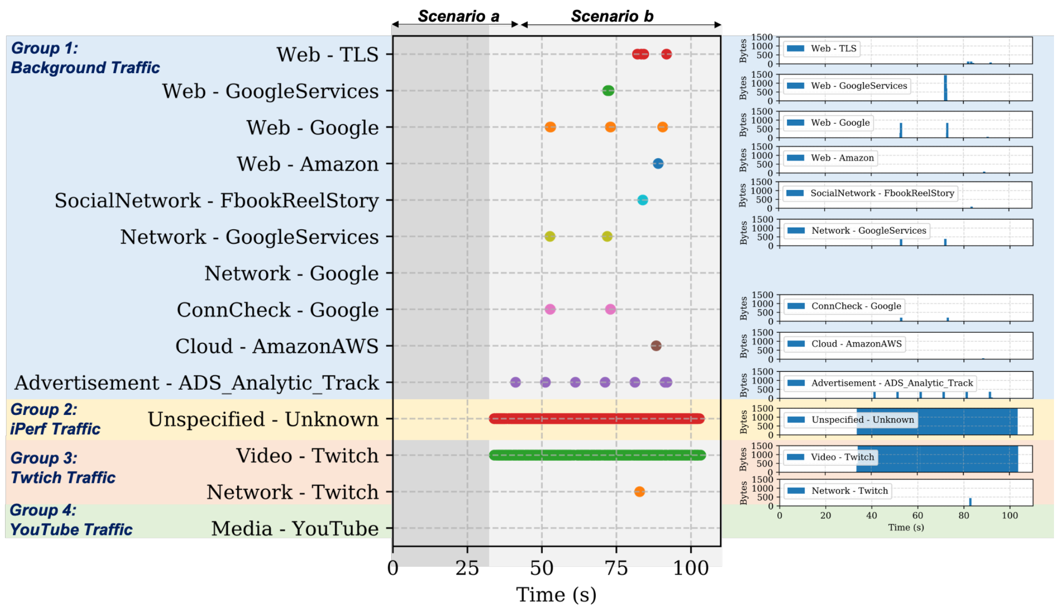
UE	Radio Resource Scheduler	Traffic Management Pipeline		
	Slice Scheduling Rules	**Classifier Rules**	**Queuing Rules**	**Shaper Rules**
UE 1	**Slice 1**	Rule 1: Unknown TCP traffic (User App1: iPerf)	Rule 1: FIFO	Rule 1: 10 Mbps
	Bitrate: 50 Mbps	Rule 2: Live video streaming traffic (User App2: Twitch)	Rule 2: CoDel	Rule 2: 40 Mbps
	Deadline: 2.5 ms	Rule 3: Other traffic (User App2: Web, Advertisement ...)	Rule 3: CoDel	Rule 3: 10 Mbps
UE 2	**Slice 2**	Rule 4: Unknown TCP traffic (User App3: iPerf)	Rule 4: FIFO	Rule 4: 10 Mbps
	Bitrate: 120 Mbps	Rule 5: Video streaming traffic (User App4: YouTube)	Rule 5: FIFO	Rule 5: 120 Mbps
	Deadline: 5 ms	Rule 6: Other traffic (User App4: Web, Advertisement ...)	Rule 6: FIFO	Rule 6: 10 Mbps

Network Configurations. The gNB operates on band 41 with 20 MHz bandwidth and a TDD frame of 7 downlink slots, 2 uplink slots, and 1 mixed slot. To minimize latency, the NearRT-RIC and xApp run on the same machine as the gNB. Two Pixel phones (UEs) connect to the gNB over-the-air, as shown in Figure 4.10.

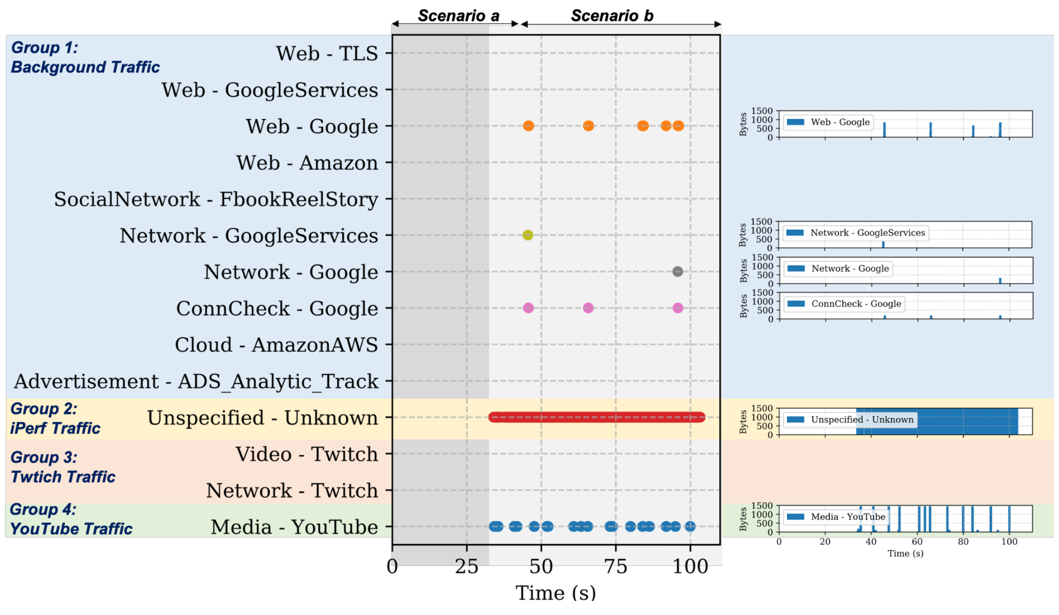
Slice and Traffic Policies. The AUTO-RAN xApp implements a set of predefined rule-based policies that enable an autonomous control loop. Upon detecting application-specific traffic (awareness and analysis), the xApp identifies the relevant rules (decision-making) and generates control messages (control execution) to be applied at the gNB. These policies fall into two categories: traffic control and slice control, as summarized in Table 4.4. The details of the traffic control policies are elaborated below:

- For iPerf traffic (labeled as unknown) in both UEs, the xApp applies a conservative policy to prevent it from monopolizing bandwidth. It creats a classifier rule to isolate the traffic, assigns a FIFO queue for basic scheduling, and sets a shaper to cap the bandwidth at maxiumum 10 Mbps.
- For Twitch traffic in UE 1, which represents live video streaming and is highly sensitive to latency and jitter, the xApp applies a classifier rule, assigns a CoDel queue (with a 100 ms target delay) to actively manage latency, and configures a shaper to allocate 40 Mbps, ensuring smooth, real-time streaming performance.
- For YouTube traffic in UE 2, which is bursty and more tolerant to delay, the xApp uses a classifier rule, assigns a FIFO queue, and configures a shaper to allocate 120 Mbps. This higher bandwidth supports efficient buffering during bursts without affecting other traffic.
- Background traffic, such as web services and advertisements, is treated as low priority. For both UEs, the xApp applies a classifier rule and sets the bandwidth limit to 10 Mbps. However, queuing policies differ by service type: UE 2 (eMBB) uses a FIFO queue, while UE 1, which supports uRLLC, employs a CoDel queue to reduce latency even for background flows.

While applying traffic control rules, the xApp applies slice control policies at the gNB. It creates two slices: Slice 1 for UE 1, optimized for uRLLC services with tighter scheduling deadlines, and Slice 2 for UE 2, configured for eMBB services with greater bandwidth allocation. This design ensures that each UE receives resources aligned with its application requirements and QoS demands.



(a) UE 1.



(b) UE 2.

Figure 4.11: Timeline of packet reception (left) and data volume (right) per application protocol for both UEs.

Traffic Analysis. Figure 4.11 presents the detailed traffic information received by the gNB for each UE and compares two scenarios: Scenario a (without AUTO-RAN xApp) and Scenario b (with AUTO-RAN xApp activated at 33 seconds). In Scenario b, the deployed xApp gains access to fine-grained packet-level data, including Layer 7 protocol information, enabling it to classify traffic within each UE. In contrast, Scenario a lacks this capability, limiting traffic visibility within gNB. In Scenario b, each packet is analyzed using the nDPI library in gNB and reported to the xApp, which classifies traffic into four categories: (1) background traffic, (2) iPerf traffic[9], (3) Twitch traffic, and (4) YouTube traffic. The right side of Figure 4.11 shows the data volume of each arrived packet over the time. Among them, iPerf and Twitch traffic exhibit the highest arrival rates (approximately 850 packets per second) and are transmitted using the MTU size of 1500 bytes. In contrast, YouTube traffic, as observed for UE 2 in Figure 4.11b, follows a bursty transmission pattern, sending groups of 500 to 1200 packets roughly every 5 seconds. This behavior is distinct from the continuous flow seen in live Twitch video or iPerf traffic. Meanwhile, low-frequency packets such as web service requests, connection checks, and advertisements are classified as background traffic due to their infrequent arrival.

Network and User Performance. Figure 4.12 evaluates both network- and user-level performance metrics across two scenarios. The metrics include RB utilization, bitrate, RLC buffer delay, and the average latency of each control message.

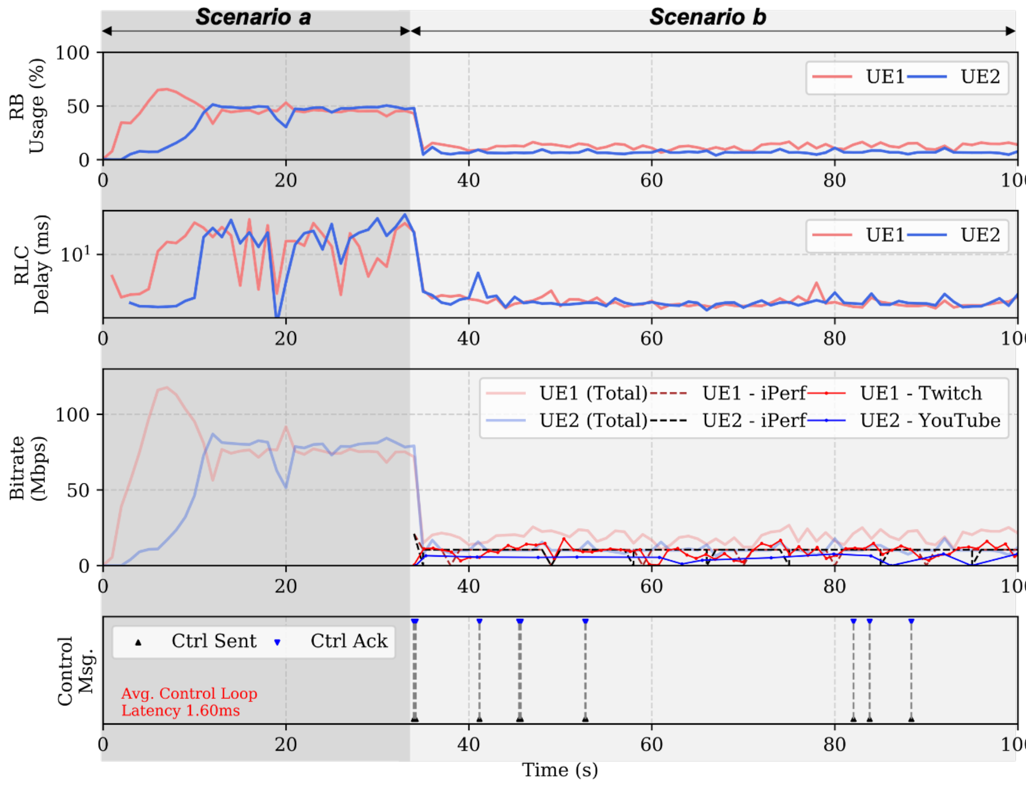


Figure 4.12: Measurement of network and user performance.

[9]iPerf lacks Layer 7 signatures and is marked as unknown by nDPI.

(1) Scenario a: From 0 to 33 s, both UEs show approximately 50% RB usage. This is primarily due to iPerf traffic consuming a large portion of the bandwidth, which in turn leads to elevated RLC buffer delays[10] (up to 100 ms) for both UEs. The bitrate shown in this phase represents the total bitrate, without visibility into individual application flows.

(2) Scenario b: Starting at 34 s with AUTO-RAN xApp activated, RB usage drops significantly: to around 12% for UE 1 and 6% for UE 2. This reduction is primarily due to traffic shaping applied to iPerf traffic, which is capped at 10 Mbps. UE 1 also runs live video streaming, which requires continuous data transmission and results in slightly higher RB usage than UE 2, which streams buffered video with a bursty pattern. Following the xApp deployment, RLC buffer delay drops sharply to approximately 0.5 ms, mainly due to the pacer[11] introduced by the traffic control mechanism (as detailed in Section 3.4). The pacer regulates packet flow from upper layers to the RLC buffer, helping prevent buffer bloat and reducing delay. Additionally, per-application bitrate statistics become available, as the TC SM enables the xApp to identify and monitor individual traffic flows, providing finer-grained visibility for network operators. Results show that iPerf maintains a steady 10 Mbps, in line with the shaping policy. YouTube exhibits a lower bitrate ($\approx$5 Mbps) compared to Twitch ($\approx$10 Mbps), reflecting its bursty traffic behavior versus Twitch's continuous stream. Finally, the average control loop latency is measured at 1.6 ms, including both traffic and slice control messages. Traffic control messages are triggered upon detection of new traffic types. Slice control is executed only once - when the xApp detects two UEs with different service types, it sends two control messages to create slices and associate each UE accordingly. The slice configuration then persists at the gNB.

Control Latency Breakdown. We present a detailed latency breakdown, including monitoring, control logic, and end-to-end control loop latencies, as illustrated in Figure 4.13, with measurement results shown in Figure 4.14.

(1) Monitoring and Control Loop Latencies: The P99 monitoring latency is approximately 1.27 ms across all traffic types, with low variance, highlighting the xApp's stable and responsive monitoring of the RAN. In contrast, the control loop latency varies across traffic types due to the different control policies applied. Specifically, traffic types requiring the creation of FIFO queues experience higher latency. This is because the traffic control mechanism within the RAN leverages shared libraries to dynamically load queue implementations. Loading and instantiating a FIFO queue takes approximately 1 ms, while a CoDel queue can be created in under 50 µs. For instance, UE 1's Twitch and background traffic are managed using CoDel queue policies, resulting in much lower latencies. As a result, the P99 control loop latency is around 0.4 ms for traffic types using CoDel queues, and can reach up to 11 ms for those using FIFO queues. Additionally, the slice control latency achieves 0.31 ms, demonstrating the xApp's ability to real-time configure and assign slices efficiently.

[10]RLC buffer delay is the time a packet remains in the buffer. Each UE has one RLC buffer, as only a single DRB is configured.

[11]The pacer is initialized when the xApp is deployed.

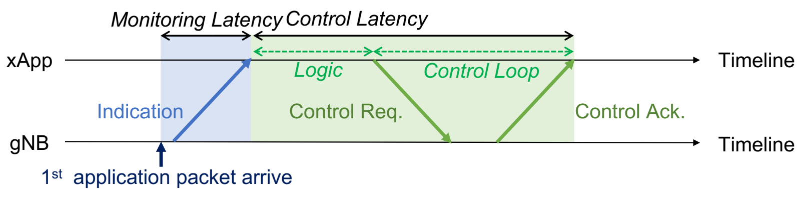
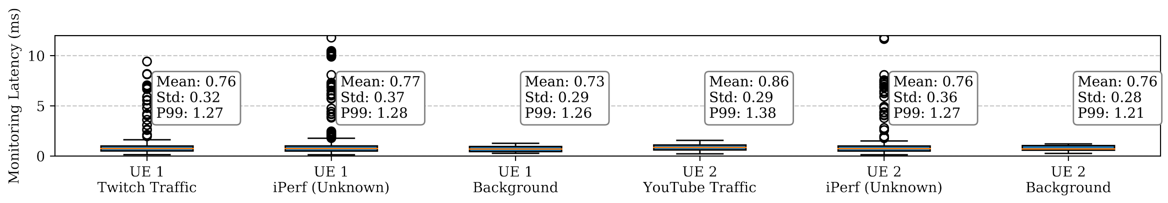
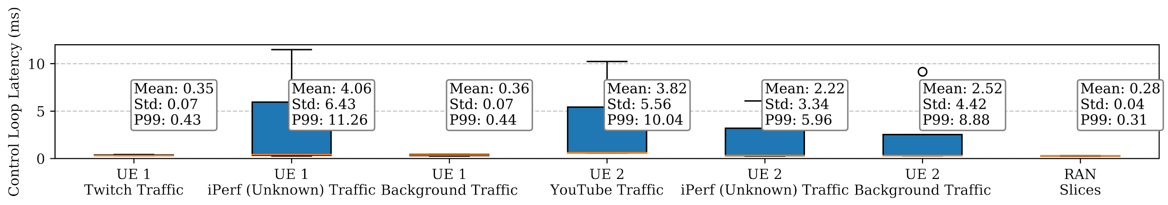


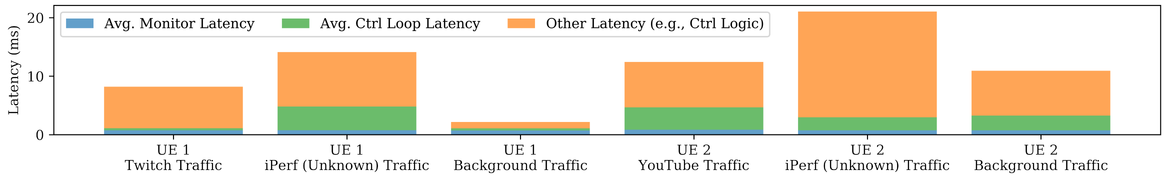
Figure 4.13: Latency breakdown: From gNB reception of the first application packet to xApp receiving the monitoring data and completing control execution for specific traffic types.



(a) Monitor latency of each type of traffic.



(b) Control loop latency of traffic and slice control.



(c) Total latency including control logic processing time within xApp.

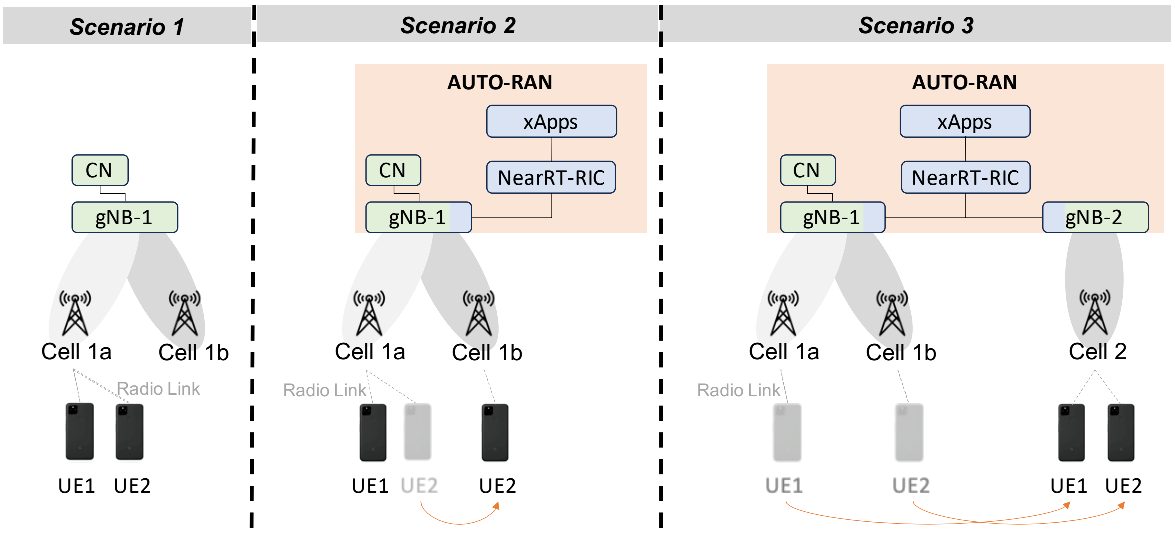
Figure 4.14: Latency breakdown for controlling different traffic types.

(2) Control Logic Latency: The control logic latency within the xApp is further derived by subtracting the average monitoring and control loop latencies from the total latency, as shown in Figure 4.14c. Here, total latency is defined as the time interval between the reception of the first application packet at the gNB and the completion of the corresponding control action for that traffic type. The variation in control logic latency (ranging from under 1 ms to up to 15 ms) stems from the internal thread design of the AUTO-RAN xApp, which currently lacks support for parallel execution of control policies (see Figure 4.7). When multiple traffic types trigger control decisions simultaneously, thread contention occurs, leading to latency spikes (UE 2 iPerf traffic) in our measurements. This highlights the need for design improvements, particularly to enable concurrent policy handling. Moreover, incorporating more advanced techniques such as ML-based decision models could further increase control latency, underscoring a key trade-off: while a more intelligent or feature-rich xApp may enhance decision quality, it can also add computational overhead that challenges real-time adaptability.

Finally, this experiment demonstrates that the proposed AUTO-RAN xApp can autonomously optimize network behavior without requiring direct operator interaction with O-RAN control mechanisms. Once deployed, the xApp dynamically applies traffic and slice control policies based on current network conditions, thereby streamlining the performance optimization process for network operators.

4.5.2 Enabled Network-Driven Mobility Management

Setup and Workload. To demonstrate the real-time network adaptability of AUTO-RAN through the dynamic handover control , we set up three scenarios, as shown in Figure 4.15, to compare network and user performance with and without AUTO-RAN. Scenario 1 serves as the baseline without AUTO-RAN, while Scenarios 2 and 3 incorporate AUTO-RAN for dynamic network optimization. This experiment focuses on two key use cases that benefit from automated handover control: load balancing and energy saving. In the load balancing use case (Scenario 2), when both UEs are connected to the same cell, RB usage reaches 100%, causing potential congestion. To alleviate this, the operator can leverage intra-handover control to move UE 2 from Cell 1a to Cell 1b in gNB-1, redistributing the load while maintaining service quality. In the energy-saving use case (Scenario 3), the operator aims to shut down a high-powered gNB during low-traffic periods to reduce energy consumption. To ensure seamless service continuity, the operator initiates inter-handover control, transferring all UEs from gNB-1 to gNB-2, which operates with lower power and resource usage. During scenario transitions, each UE continuously receives a TCP traffic flow, generated by iPerf from the CN. Network and user performance metrics — including bitrate, RB usage, and cell states — are collected by the monitoring xApp (as illustrated in Figure 4.8) using KPM and RC SMs. Additionally, control loop latency is measured within the control xApp, as depicted in Figure 4.8.



Components	CN, gNB-1, gNB-2	UE1, UE2	NearRT-RIC, xApps
Platform/Hardware Used	Amarisoft Callbox Ultimate	Pixel 5	FlexRIC

Figure 4.15: Experiment setup for mobility scenarios with and without AUTO-RAN.

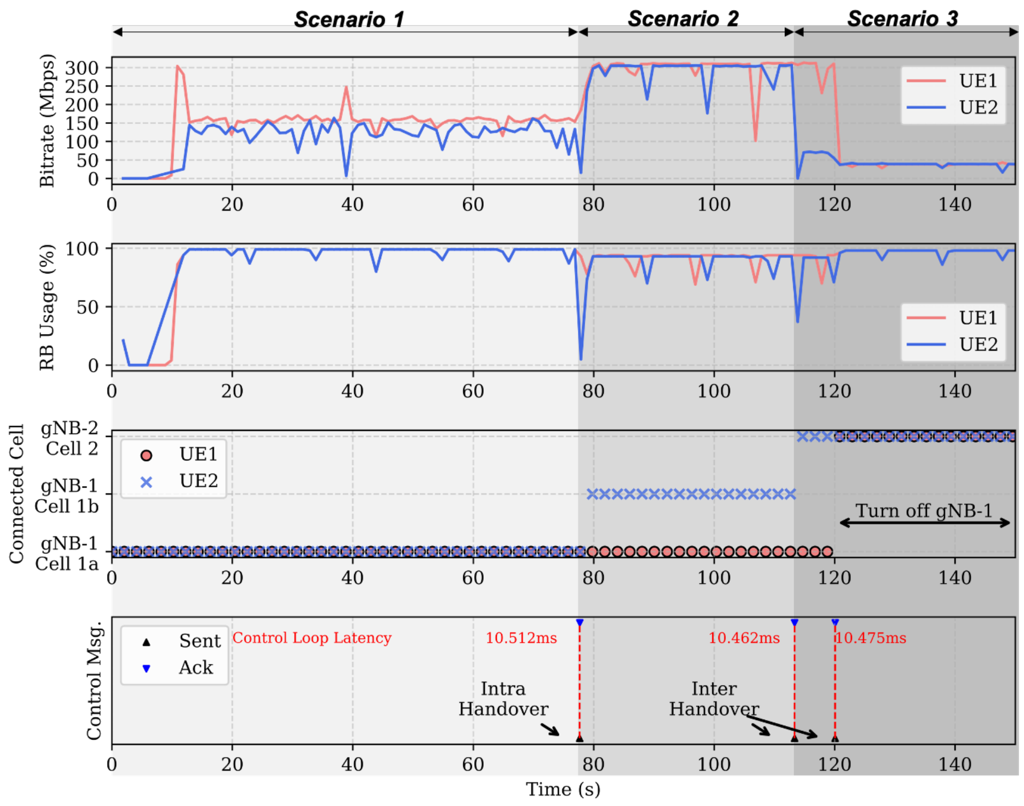


Figure 4.16: Measurement of network and user statistics across scenarios and evaluation of handover control loop latency.

Network Configurations. In our experimental setup, gNB-1 is configured on band 78 with Cell 1a (40 MHz) and Cell 1b (40 MHz), while gNB-2 operates on band 41 with a single 20 MHz cell. The CN and gNBs run on the Amarisoft Callbox, and NearRT-RIC and xApps are deployed on the same machine to minimize network latency when controlling the gNBs. Additionally, two Pixel phones (UEs) are connected to gNB-1/gNB-2 over-the-air by placing them, along with the output antenna of gNB-1/gNB-2, inside a Faraday cage to isolate the test environment.

Load Balancing. In Scenario 1, as shown in Figure 4.16, from 0s to 77s, both UEs are connected to Cell 1a in gNB-1, causing RB usage to reach 100%. At 78s (Scenario 2), the xApp sends a control message to initiate intra-handover, transferring UE 2 to Cell 1b. After the handover, the bitrate of each UE increases from 150 Mbps to 300 Mbps since each UE can now utilize the full 40 MHz bandwidth of its respective cell. Additionally, RB usage decreases to approximately 95%, likely due to more efficient resource allocation and reduced contention within each cell. The cell state information updates with a 2-second delay compared to the bitrate and RB usage statistics, as cell information is refreshed every 2 seconds.

Energy Saving. At 115s (Scenario 3), the xApp sends an inter-handover control message to gNB-1, instructing it to move UE 2 to gNB-2. At 120s, the xApp issues a second control message, this time moving UE 1 to gNB-2. Once all UEs have been successfully transferred, the xApp sends a final control message to reduce the transmission gain of gNB-1 to zero, effectively simulating a power-off state[12]. By 120s, both UEs are connected to gNB-2, with bitrate reduced to approximately 50 Mbps due to the 20MHz bandwidth limitation of gNB-2. This transition ensures seamless service continuity by actively managing the handover process, transferring UEs to another gNB before shutting down while maintaining service availability.

Control and Indication Message Latencies. The average control loop latency is approximately 10 ms, primarily due to the delay between the proxy agent and the WebSocket server of the gNB, where the use of HTTP services adds additional latency. The latency for generating the control message is minimal, taking only 14 µs. The latency for transmitting the RC handover control message between the xApp and the proxy agent is around 2 ms, while the remaining 8 ms of latency occurs between the proxy agent and WebSocket server of gNB. Additionally, the indication message, used for real-time data collection, is encapsulated in the KPM format and has an average one-way latency of approximately 0.442 ms, demonstrating its capability to monitor and analyze network conditions in real time.

4.6 Discussion and Future Work

The proposed AUTO-RAN architecture enables operators to shift from imperative to declarative control of mobile networks, significantly reducing operational burden. While this approach simplifies interaction with the system, the introduction of an autonomous control loop opens a wide range of research opportunities in the area of self-optimizing and self-adaptive RAN architectures.

For example, the analysis step within the proposed autonomous control loop could be enhanced with ML models to predict potential network issues, such as congestion during peak hours or signaling storms. The decision-making step could be augmented with emerging technologies like digital twins, allowing the system to simulate and evaluate possible control actions before execution. Moreover, Large Language Models (LLMs) could be integrated to allow operators to define policies in natural language, which would then be translated into machine-executable control logic—further lowering the barrier to managing complex mobile infrastructures.

However, integrating such an autonomous mechanism within the O-RAN architecture presents several challenges that require careful consideration. A primary concern is the lack of standardized communication mechanisms between control applications, such as between xApps or between rApps. Unlike the A1 interface between the NonRT-RIC and xApps, interfaces for peer-to-peer coordination among control applications remain underspecified. As networks adopt more modular and hierarchical control logic, relying on a

[12]This strategy can be further extended by leveraging OAM/SMO to completely shut down gNB-1 by undeploying the network, rather than just decreasing the transmission gain.

single monolithic application becomes insufficient. This calls for either the introduction of an upper-layer orchestration framework to manage control logic distributed across applications, or the development of new protocols or mechanisms to support inter-application coordination and conflict resolution.

Another critical issue is the uncertainty surrounding control latency in different deployment scenarios. AUTO-RAN offers the flexibility to distribute the four key steps of the autonomous control loop across multiple applications and time domains, based on specific use case requirements. For instance, tightly coupled deployment near E2-nodes can reduce latency, but embedding advanced logic such as ML models may introduce processing overhead that compromises real-time responsiveness. Distributing these steps across separate applications and time domains may help mitigate this, but it introduces coordination complexity, further reinforcing the need for standardized inter-application communication.

These challenges highlight the importance of future research on control orchestration and latency-aware deployment strategies, both of which are critical for advancing programmable and autonomous RAN architectures.

4.7 Conclusions

In this chapter, we introduce AUTO-RAN, a novel concept that enables autonomous programmability through a robust SD-RAN application design for next-generation mobile networks. AUTO-RAN extends the existing O-RAN architecture by incorporating an autonomous control loop that abstracts underlying complexity of O-RAN and supports declarative interactions for network operators. This innovation simplifies network optimization and significantly reduces operational overhead. AUTO-RAN offers several key benefits: it hides the underlying complexity of O-RAN, supports real-time adaptability to dynamic network conditions, and enables seamless integration of new features and advancements from both 3GPP and O-RAN ecosystems. Its feasibility has been demonstrated through two representative use cases, namely RAN slicing and mobility management, highlighting its ability to autonomously optimize both network and user performance in real time.

Chapter 5

Conclusion

5.1 Summary

The evolution of 5G and beyond has shifted the mobile network paradigm toward openness, programmability, and application-centric design. These trends aim to reduce vendor lock-in and enable more flexible, customizable deployments that can adapt to diverse service requirements and dynamic network conditions. In this context, the concept of Open RAN (particularly through the efforts of the O-RAN Alliance) plays a crucial role by introducing a disaggregated architecture centered around the RIC, which supports both non-real-time (rApps) and near-real-time (xApps) control applications. This architecture empowers vendors and operators to tailor deployments and implement intelligent control mechanisms, unlocking new opportunities for innovation and network optimization.

Despite these advancements, the shift toward open and programmable RAN architectures introduces significant challenges. As mobile networks continue to evolve to support emerging applications such as AR/VR and haptic communication, the need for greater control flexibility, unified coordination of UP functions, and scalable system integration becomes increasingly critical. However, centralized control mechanisms introduced by SD-RAN and O-RAN often constrain vendors and operators from customizing their networks due to limited standardization and platform-specific implementations. This also makes x/rApps more difficult to develop, port, and reuse. In addition, the 5G RAN, often functioning as a “fat Layer 2,” creates disparities between IP flow control and radio resource allocation. The integration of O-RAN technologies into existing mobile infrastructures further compounds this complexity. Operators face increasing operational overhead due to fragmented control domains defined separately by 3GPP and O-RAN.

In light of these challenges, this thesis explores the design of a programmable RAN architecture for next-generation mobile networks, with the goal of enhancing flexibility and programmability across both the control and user planes, while simplifying network infrastructure. Our contributions are organized into three key areas:

Flexible Control Plane. In Chapter 2, we introduced the FlexCtrl architecture, which supports three control plane topologies (including centralized, decentralized, and distributed) by evolving the control capabilities of both O-RAN and SD-RAN systems to address the challenges of control flexibility and real-time programmability. Within the decentralized topology of FlexCtrl, the FlexApp framework is proposed to further address the complexity of xApp development in O-RAN. FlexApp mitigates NearRT-RIC platform lock-in and improves xApp reusability through the introduction of the E2* interface, which reduces both latency and CPU utilization, thereby enhancing portability and enabling ultra-low-latency control operations. To validate the real-time programmability offered by FlexCtrl, we redesign the radio resource scheduler to support recursive operations, facilitating virtualization and multi-level control. Prototype evaluations demonstrate that FlexApp, through the proposed E2* interface, achieves ultra-low latency (less than 10 ms), high scalability, and minimal overhead in xApp operation, while also supporting two-level abstraction to simplify xApp development. Likewise, the FlexCtrl prototype confirms its ability to support flexible control topologies and real-time programmability for dynamic RAN slicing.

Our evaluation of FlexCtrl shows that customizable placement of control logic across centralized, decentralized, and distributed topologies is feasible in practice. It maintains ultra-low-latency operation even as the number of xApps increases and during the recursive deployment of two-level abstract control logic, suggesting its potential for hierarchical control deployment in future intent-driven mobile networks.

Integrated and Programmable User Plane. In Chapter 3, we introduced IUP, an innovative RAN system that integrates UPF functionalities into the RAN, serving as a foundational component for next-generation mobile networks. IUP addresses two major challenges: the lack of coordination between UP functions across the CN and RAN, and the increasing complexity of distributed UP functions. To this end, we propose the IDFC sublayer, positioned above the SDAP layer, which introduces a new traffic management pipeline along with programmable rules for both IP traffic control and radio resource allocation. IUP reduces N3-related overhead and processing costs while simplifying the end-to-end data delivery path. It also enhances UP programmability by extending control from the IP layer to the radio link layer, providing a unified framework for managing packet flows and radio resources. Furthermore, IUP enables universal connectivity by operating as a Layer 3 device, seamlessly integrating diverse access technologies through the IP protocol. We also analyze IUP's implications for key mobility scenarios such as handover and roaming, including its interaction with existing CP functions to support backward compatibility and RAN disaggregation. Finally, prototype evaluations demonstrate that IUP reduces latency and overhead by up to 50%, supports seamless integration of 3GPP and non-3GPP networks, and enables real-time programmability of both traffic control and resource allocation.

Prototype evaluation of IUP reveals that coordinating IP flow control and radio resource allocation within the RAN node is practically realizable, without introducing signaling overhead or latency spikes. The implementation also confirms the viability of seamless IP-based convergence across heterogeneous access technologies, suggesting that IUP could support future real-time applications with strict performance requirements and promote broader adoption and integration of 3GPP networks.

Autonomous Radio Acces Network. In Chapter 4, building on the foundations established by FlexCtrl and IUP, we introduced AUTO-RAN, a novel concept that abstracts the complexities of O-RAN and enables autonomous programmability through a robust SD-RAN application design. AUTO-RAN addresses the growing complexities of managing mobile network infrastructures that span both 3GPP and O-RAN ecosystems. It simplifies network optimization process and reduces operational overhead through an autonomous control loop, allowing operators to interact with the network in a declarative manner, eliminating the need to manage the internal complexity of O-RAN technologies directly. AUTO-RAN also facilitates the seamless integration of evolving standards and capabilities from both 3GPP and O-RAN ecosystems, providing a future-proof foundation for future mobile networks. We further discuss the lifecycle of control applications in AUTO-RAN and present examples demonstrating how they evolve and abstract O-RAN complexities to support high-level, intent-based network control. Finally, the prototype of AUTO-RAN showcases its real-time adaptability through two key use cases. First, in the automated RAN slicing use case, the system effectively handles diverse application flows and maintains average latencies below 1 ms for both traffic monitoring and slice control. In the mobility management use case, AUTO-RAN demonstrates its ability to handle declarative operator intents for load balancing and energy-saving scenarios, performing handover control while ensuring service continuity.

Prototype evaluation of AUTO-RAN confirms that autonomous, declarative control of mobile networks spanning both 3GPP and O-RAN ecosystems is practically achievable. The system reduces x/rApp development complexity, maintains sub-1 ms latency for traffic monitoring and slice control, and demonstrates real-time adaptability to dynamic network conditions, indicating its potential for future self-optimizing and AI-native mobile networks.

In addition to the main contributions presented above, this thesis includes several demonstrations (summarized in Table 1.2) as well as various technical developments. Specifically, the maintenance of the FlexRIC[1], its integration with srsRAN[2], and the implementation of RAN slicing in OAI[3] have enabled numerous researchers to build and extend their work in the context of 5G and Open RAN [55, 58, 63, 105, 106].

5.2 Future Work

A programmable RAN architecture offers significant potential for future exploration, it is enabled by the proposed flexible control plane (FlexCtrl), integrated and programmable user plane (IUP), and autonomous control mechanisms (AUTO-RAN). These building blocks aim to deliver customizable, fine-grained, and automated programmability across both control and user planes. This not only enables real-time network and user performance optimization but also supports autonomous operation, helping to reduce complexity for network operators. Future work related to each contribution has been discussed individually within their respective chapters (see Sections 2.7, 3.8, and 4.6). In the following, we focus on high-level directions for future research.

As emerging applications become more diverse and complex, and the number of connected devices continues to grow, new requirements are pushing beyond the capabilities of the original usage scenarios defined in IMT-2020 (5G), namely eMBB, uRLLC, and mMTC. To address these limitations, IMT-2030 (6G) introduces three new usage scenarios: ubiquitous connectivity, AI-native communications, and Integrated Sensing and Communication (ISAC) [130]. Among them, ubiquitous connectivity aims to provide seamless and continuous access to communication networks across all environments (e.g., urban, rural, remote, airborne, maritime, and underwater), including underserved and hard-to-reach areas. Achieving this goal will require the integration of diverse wired and wireless technologies and ensuring reliable device connections under varying network conditions.

[1]https://gitlab.eurecom.fr/mosaic5g/flexric/-/tree/br-flexric?ref_type=heads
[2]https://docs.srsran.com/projects/project/en/latest/tutorials/source/near-rt-ric/source/index.html
[3]https://gitlab.eurecom.fr/oai/openairinterface5g/-/merge_requests/2458

Multi-RAT Interworking. In this context, the proposed IUP is a promising direction for future network infrastructure. By incorporating UPF functionalities directly within the RAN node, IUP acts as a Layer 3 device, enabling flexible, IP-based interworking across multiple access technologies (as shown in Figure 3.5). This not only simplifies the integration process for heterogeneous networks but also facilitates their convergence. Moreover, the deployment of IUP in technologies such as Unmanned Aerial Vehicles (UAVs) and Non-Terrestrial Networks (NTNs) could enhance network resiliency during natural disasters by enabling local communication within isolated regions without relying on external networks. However, to fully realize these benefits, further research is needed. In particular, efficient data convergence and routing mechanisms must be developed for scenarios where devices simultaneously connect to multiple access technologies (e.g., 5G and Wi-Fi) to support collaborative networking [27]. As IUP introduces programmability from the IP layer down to the radio link, future research should explore methods for coordinating data flows and managing radio resources across heterogeneous technologies to optimize overall performance and user experience [93]. Furthermore, the role of IPv6 deserves attention [143], as its advanced addressing and routing capabilities are fundamental for enabling large-scale, flexible, and programmable networking in IUP-enabled environments.

Intent-Driven and AI-Native Control. While the previous discussion centers on the UP, future research in the CP domain will be equally essential to enable intelligent and adaptive RAN architectures. Building on the modular foundation provided by FlexCtrl and AUTO-RAN, there are promising opportunities to develop intent-driven CP frameworks, where high-level service goals can be dynamically translated into low-level control policies. This also opens avenues for designing telecom-grade APIs that support the plug-and-play integration of new control applications, aligning with the goal of achieving network operational simplicity [93]. In addition, the rise of AI-native communications and ISAC introduces new demands on the CP to make intelligent, real-time decisions based on dynamic network conditions. Future work may explore the integration of digital twin frameworks, multi-agent reinforcement learning, and graph-based control representations to support more autonomous and predictive control. There is also potential in investigating cross-domain orchestration, where control decisions span not only 3GPP networks, but also non-3GPP access, edge computing platforms, and application-layer services. Finally, these research directions lay the foundation for a new era of intelligent and adaptive RAN architectures, bringing the vision of 6G networks closer to reality.

Reference

[1] 3GPP. *5G; Access to the 3GPP 5G Core Network (5GCN) via non-3GPP access networks (3GPP TS 24.502 version 17.9.0 Release 17).* Tech. rep. ETSI TS 124 502 V17.9.0. 3GPP, July 2023.

[2] 3GPP. *5G; Architecture for enabling Edge Applications (3GPP TS 23.558 version 17.10.0 Release 17).* Tech. rep. ETSI TS 123 558 V17.10.0. 3GPP, Jan. 2024.

[3] 3GPP. *5G; Management and orchestration; 5G end to end Key Performance Indicators (KPI) (3GPP TS 28.554 version 18.7.0 Release 18); Management and orchestration; 5G end to end Key Performance Indicators (KPI).* Tech. rep. ETSI TS 128 554 V18.7.0. 3GPP, Oct. 2024.

[4] 3GPP. *5G; Management and orchestration; Architecture framework (3GPP TS 28.533 version 18.3.0 Release 18).* Tech. rep. ETSI TS 128 533 V18.3.0. 3GPP, Oct. 2024.

[5] 3GPP. *5G; Management and orchestration; Concepts, use cases and requirements (3GPP TS 28.530 version 17.4.0 Release 17).* Tech. rep. ETSI TS 128 530 V17.4.0. 3GPP, Apr. 2024.

[6] 3GPP. *5G; Management and orchestration; Levels of autonomous network (3GPP TS 28.100 version 18.0.0 Release 18).* Tech. rep. ETSI TS 128 100 V18.0.0. 3GPP, May 2024.

[7] 3GPP. *5G; Management and orchestration; Self-Organizing Networks (SON) for 5G networks (3GPP TS 28.313 version 18.2.0 Release 18).* Tech. rep. ETSI TS 128 313 V18.2.0. 3GPP, Oct. 2024.

[8] 3GPP. *5G; NG-RAN; Architecture description (3GPP TS 38.401 version 18.3.0 Release 18).* Tech. rep. ETSI TS 138 401 V18.3.0. 3GPP, Sept. 2024.

[9] 3GPP. *5G; NG-RAN; Architecture description (3GPP TS 38.401 version 18.3.0 Release 18).* Tech. rep. ETSI TS 138 401 V18.3.0. 3GPP, Sept. 2024.

[10] 3GPP. *5G; NR; Medium Access Control (MAC) protocol specification (3GPP TS 38.321 version 18.3.0 Release 18).* Tech. rep. ETSI TS 138 321 V18.3.0. 3GPP, Oct. 2024.

[11] 3GPP. *5G; NR; NR and NG-RAN Overall description; Stage-2 (3GPP TS 38.300 version 18.2.0 Release 18)).* Tech. rep. ETSI TS 138 300 V18.2.0. 3GPP, Aug. 2024.

[12] 3GPP. *5G; NR; Packet Data Convergence Protocol (PDCP) specification (3GPP TS 38.323 version 18.3.0 Release 18).* Tech. rep. ETSI TS 138 323 V18.3.0. 3GPP, Oct. 2024.

[13] 3GPP. *5G; NR; Radio Link Control (RLC) protocol specification (3GPP TS 38.322 version 18.1.0 Release 18).* Tech. rep. ETSI TS 138 322 V18.1.0. 3GPP, Aug. 2024.

[14] 3GPP. *5G; NR; Radio Resource Control (RRC); Protocol specification (3GPP TS 38.331 version 18.3.0 Release 18).* Tech. rep. ETSI TS 138 331 V18.3.0. 3GPP, Oct. 2024.

[15] 3GPP. *5G; Procedures for the 5G System (5GS) (3GPP TS 23.502 version 18.7.0 Release 18).* Tech. rep. ETSI TS 123 502 V18.7.0. 3GPP, Oct. 2024.

[16] 3GPP. *5G; Study on Artificial Intelligence/Machine Learning (AI/ ML) management (3GPP TR 28.908 version 18.1.0 Release 18).* Tech. rep. ETSI TR 128 908 V18.1.0. 3GPP, Oct. 2024.

[17] 3GPP. *5G; Study on enhancements for Artificial Intelligence (AI)/Machine Learning (ML) for NG-RAN (Release 19).* Tech. rep. 3GPP TR 38.743 V19.0.0. 3GPP, Sept. 2024.

[18] 3GPP. *5G; System architecture for the 5G System (5GS) (3GPP TS 23.501 version 18.5.0 Release 18).* Tech. rep. ETSI TS 123 501 V18.5.0. 3GPP, May 2024.

[19] 3GPP. *Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; 5G; IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3 (3GPP TS 24.229 version 18.6.0 Release 18).* Tech. rep. ETSI TS 124 229 V18.6.0. 3GPP, Oct. 2024.

[20] 3GPP. *LTE; 5G; Interface between the Control Plane and the User Plane nodes (3GPP TS 29.244 version 18.5.0 Release 18).* Tech. rep. ETSI TS 129 244 V18.5.0. 3GPP, May 2024.

[21] 3GPP. *LTE; 5G; Management and orchestration; Management services for communication service assurance; Requirements (3GPP TS 28.535 version 18.2.0 Release 18).* Tech. rep. ETSI TS 128 535 V18.2.0. 3GPP, Oct. 2024.

[22] 3GPP. *Management and orchestration; Study on management aspect of Network Digital Twin (Release 19).* Tech. rep. 3GPP TR 28.915 V2.0.0. 3GPP, Dec. 2024.

[23] 3GPP. *NR; Study on XR enhancements for NR (3GPP TS 38.321 version 18.3.0 Release 18).* Tech. rep. ETSI TS 138 321 V18.0.1. 3GPP, Apr. 2023.

[24] 3GPP. *Study on Key Quality Indicators (KQIs) for 5G service experience (Release 18).* Tech. rep. 3GPP TR 28.863 V18.0.0. 3GPP, Dec. 2023.

[25] 3GPP. *Study on management aspects of NTN – Phase 2 (Release 19).* Tech. rep. 3GPP TR 28.874 V2.0.0. 3GPP, Dec. 2024.

[26] 3GPP. *Universal Mobile Telecommunications System (UMTS); LTE; Telecommunication management; Self-Organizing Networks (SON); Concepts and requirements (3GPP TS 32.500 version 18.0.0 Release 18).* Tech. rep. ETSI TS 132 500 V18.0.0. 3GPP, May 2024.

[27] Bengt Ahlgren et al. "A survey of information-centric networking". In: *IEEE Communications Magazine* 50.7 (2012), pp. 26–36. DOI: 10.1109/MCOM.2012.6231276.

[28] AI-RAN Alliance. *AI-RAN Alliance Vision and Mission White Paper.* Dec. 2024. URL: https://ai-ran.org/publications/.

[29] O-RAN Alliance. *A1 interface: General Aspects and Principles.* Tech. rep. Technical Specification O-RAN.WG2.A1GAP-R004-v04.00. O-RAN Alliance, Oct. 2024.

[30] O-RAN Alliance. *Non-RT RIC: Architecture.* Tech. rep. Technical Specification O-RAN.WG2.Non-RT-RIC-ARCH-R004-v06.00. O-RAN Alliance, Oct. 2024.

[31] O-RAN Alliance. *O-RAN Fronthaul Working Group; Fronthaul Interoperability Test Specification (IOT).* Tech. rep. Technical Specification O-RAN.WG4.IOT.0-R004-v12.00. O-RAN Alliance, Oct. 2024.

[32] O-RAN Alliance. *O-RAN Open F1/W1/E1/X2/Xn Interfaces Working Group; NR C-plane profile.* Tech. rep. Technical Specification O-RAN.WG5.C.1-R004-v13.00. O-RAN Alliance, Oct. 2024.

[33] O-RAN Alliance. *O-RAN Open X-haul Transport Working Group; Management interfaces for Transport Network Elements.* Tech. rep. Technical Specification O-RAN.WG9.XTRP-MGT.0-R004-v09.00. O-RAN Alliance, Oct. 2024.

[34] O-RAN Alliance. *O-RAN White Box Hardware Working Group Indoor Picocell Hardware Architecture and Requirement (FR1 Only) Specification.* Tech. rep. Technical Specification O-RAN.WG4.IOT.0-R004-v12.00. O-RAN Alliance, Oct. 2024.

[35] O-RAN Alliance. *O-RAN White Box Hardware Working Group Outdoor Macrocell Hardware Architecture and Requirements (FR1) Specification.* Tech. rep. Technical Specification O-RAN.WG7.OMAC-HAR.0-v05.00. O-RAN Alliance, Oct. 2024.

[36] O-RAN Alliance. *O-RAN WhitePaper - Building the Next Generation RAN.* Oct. 2018. URL: https://www.o-ran.org/o-ran-resources.

[37] O-RAN Alliance. *O-RAN Work Group 1 (Use Cases and Overall Architecture); SMO Intents-driven Management.* Tech. rep. Technical Report O-RAN.WG1.TR.SMO-INT-R004-v03.00. O-RAN Alliance, Oct. 2024.

[38] O-RAN Alliance. *O-RAN Work Group 10; O-RAN Operations and Maintenance Architecture.* Tech. rep. Technical Specification O-RAN.WG10.OAM-Architecture-R004-v13.00. O-RAN Alliance, Oct. 2024.

[39] O-RAN Alliance. *O-RAN Work Group 10 (OAM for O-RAN); O-RAN O1 Interface Specification.* Tech. rep. Technical Specification O-RAN.WG10.O1-Interface.0-R004-v14.00. O-RAN Alliance, Oct. 2024.

[40] O-RAN Alliance. *O-RAN Work Group 3 (Near-Real-time RAN Intelligent Controller and E2 Interface); Near-RT RIC Architecture.* Tech. rep. Technical Specification O-RAN.WG3.RICARCH-R003-v06.00. O-RAN Alliance, June 2024.

[41] O-RAN Alliance. *O-RAN Work Group 3; Near-Real-time RAN Intelligent Controller; E2 Service Model (E2SM); KPM.* Tech. rep. Technical Specification O-RAN.WG3.E2SM-KPM-R003-v05.00. O-RAN Alliance, June 2024.

[42] O-RAN Alliance. *O-RAN Work Group 3 (Near-RT RIC and E2 Interface); E2 General Aspects and Principles (E2GAP).* Tech. rep. Technical Specification O-RAN.WG3.E2GAP-R004-v06.00. O-RAN Alliance, Oct. 2024.

[43] O-RAN Alliance. *O-RAN Work Group 3 (Near-RT RIC and E2 Interface); E2 Service Model (E2SM).* Tech. rep. Technical Specification O-RAN.WG3.E2SM-R004-v06.00. O-RAN Alliance, Oct. 2024.

[44] O-RAN Alliance. *O-RAN Work Group 3 (WG-3); Near-Real-time RAN Intelligent Controller and E2 Interface; E2 Service Model (E2SM) Cell Configuration and Control.* Tech. rep. Technical Specification O-RAN.WG3.E2SM-CCC-R003-v04.00. O-RAN Alliance, June 2024.

[45] O-RAN Alliance. *O-RAN Work Group 3 (WG-3); Near-Real-time RAN Intelligent Controller and E2 Interface; E2 Service Model (E2SM), Lower Layers Control.* Tech. rep. Technical Specification O-RAN.WG3.TS.E2SM-LLC-R004-v01.00. O-RAN Alliance, Feb. 2025.

[46] O-RAN Alliance. *O-RAN Work Group 3 (WG-3); Near-Real-time RAN Intelligent Controller; E2 Service Model (E2SM), RAN Control.* Tech. rep. Technical Specification O-RAN.WG3.E2SM-RC-R003-v06.00. O-RAN Alliance, June 2024.

[47] O-RAN Alliance. *O-RAN Work Group 5 (Open F1/W1/E1/X2/Xn Interfaces Working Group); NR U-plane profile.* Tech. rep. Technical Specification O-RAN.WG5.U.0-R003-v07.00. O-RAN Alliance, June 2024.

[48] O-RAN Alliance. *O-RAN Working Group 4 (Open Fronthaul Interfaces WG); Control, User and Synchronization Plane Specification.* Tech. rep. Technical Specification O-RAN.WG4.CUS.0-R004-v16.01. O-RAN Alliance, Oct. 2024.

[49] O-RAN Alliance. *O-RAN Working Group 8 (Stack Reference Design); Stack Interoperability Test Specification.* Tech. rep. Technical Specification O-RAN.WG8.IOT.0-R004-v11.00. O-RAN Alliance, Oct. 2024.

[50] O-RAN Alliance. *R1 interface: General Aspects and Principles.* Tech. rep. Technical Specification O-RAN.WG2.R1GAP-R004-v09.00. O-RAN Alliance, Oct. 2024.

[51] AMARISOFT. *AMARI Callbox Ultimate Datasheet.* Nov. 2024. URL: https://www.amarisoft.com/test-and-measurement/device-testing/device-products/amari-callbox-ultimate.

[52] Mustafa Y. Arslan, Karthikeyan Sundaresan, and Sampath Rangarajan. "Software-defined networking in cellular radio access networks: potential and challenges". In: *IEEE Communications Magazine* 53.1 (2015), pp. 150–156. DOI: 10.1109/MCOM.2015.7010528.

[53] Arjun Balasingam, Manikanta Kotaru, and Paramvir Bahl. "Application-Level Service Assurance with 5G RAN Slicing". In: *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24).* Santa Clara, CA: USENIX Association, Apr. 2024, pp. 841–857. ISBN: 978-1-939133-39-7. URL: https://www.usenix.org/conference/nsdi24/presentation/balasingam.

[54] Luca Baldesi, Francesco Restuccia, and Tommaso Melodia. "ChARM: NextG spectrum sharing through data-driven real-time O-RAN dynamic control". In: *IEEE INFOCOM 2022-IEEE Conference on Computer Communications.* IEEE. 2022, pp. 240–249.

[55] Jorge Baranda et al. “Distributed Sequential Cloud-Native Deployment of an End-to-End 5G Network with O-RAN Functions”. In: *2024 15th International Conference on Network of the Future (NoF)*. 2024, pp. 142–144. DOI: 10.1109/NoF62948.2024.10741499.

[56] Sergio Barbarossa, Stefania Sardellitti, and Paolo Di Lorenzo. “Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks”. In: *IEEE Signal Processing Magazine* 31.6 (2014), pp. 45–55.

[57] Alessandro Bazzi et al. “On the Design of Sidelink for Cellular V2X: A Literature Review and Outlook for Future”. In: *IEEE Access* 9 (2021), pp. 97953–97980. DOI: 10.1109/ACCESS.2021.3094161.

[58] Fransiscus Asisi Bimo et al. “Design and Implementation of Next-Generation Research Platforms”. In: *2023 IEEE Globecom Workshops (GC Wkshps)*. 2023, pp. 1777–1782. DOI: 10.1109/GCWkshps58843.2023.10464635.

[59] *Border Gateway Protocol 3 (BGP-3)*. RFC 1267. Oct. 1991. DOI: 10.17487/RFC1267. URL: https://www.rfc-editor.org/info/rfc1267.

[60] Abhik Bose et al. “Leveraging Programmable Dataplanes for a High Performance 5G User Plane Function”. In: *Proceedings of the 5th Asia-Pacific Workshop on Networking*. APNet ’21. New York, NY, USA: Association for Computing Machinery, 2022, 57–64. ISBN: 9781450385879. DOI: 10.1145/3469393.3469400. URL: https://doi.org/10.1145/3469393.3469400.

[61] Pat Bosshart et al. “P4: programming protocol-independent packet processors”. In: *SIGCOMM Comput. Commun. Rev.* 44.3 (July 2014), 87–95. ISSN: 0146-4833. DOI: 10.1145/2656877.2656890. URL: https://doi.org/10.1145/2656877.2656890.

[62] Bob Briscoe et al. *Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture*. RFC 9330. Jan. 2023. DOI: 10.17487/RFC9330. URL: https://www.rfc-editor.org/info/rfc9330.

[63] Ilias Chatzistefanidis, Andrea Leone, and Navid Nikaein. “Maestro: LLM-Driven Collaborative Automation of Intent-Based 6G Networks”. In: *IEEE Networking Letters* 6.4 (2024), pp. 227–231. DOI: 10.1109/LNET.2024.3503292.

[64] Chieh-Chun Chen, Chia-Yu Chang, and Navid Nikaein. “FlexSlice: Flexible and real-time programmable RAN slicing framework”. In: *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*. 2023, pp. 3807–3812. DOI: 10.1109/GLOBECOM54140.2023.10437791.

[65] Chieh-Chun Chen, Chia-Yu Chang, and Navid Nikaein. “IUP: Integrated and Programmable User Plane for Next-Generation Mobile Networks”. In: *IEEE Network* (2025), pp. 1–1. DOI: 10.1109/MNET.2025.3551245.

[66] Chieh-Chun Chen, Alireza Mohammadi, and Navid Nikaein. "xApp independent lifecycle, Interactive RAN Slicing xApp for xApp maintainer from Open RAN operator". In: *Mobile World Congress, 27 February - 2 March 2023, Barcelona, Spain* (2023). URL: https://youtu.be/kbh31hSxVFI.

[67] Chieh-Chun Chen, Navid Nikaein, and Mikel Irazabal Bengao. "Auto-RAN: Automated application-driven RAN slicing". In: *OpenAirInterface 10th Anniversary Workshop, 12-13 September 2024, Sophia Antipolis, France* (2024). URL: https://www.eurecom.fr/publication/7860.

[68] Chieh-Chun Chen, Navid Nikaein, and Mikel Irazabal Bengoa. "FlexRIC: an SDK for next-generation SD-RANs-slicing and traffic control use-case". In: *Summer OpenAirInterface Workshop, 12-13 July 2022, Paris, France* (2022). URL: https://youtu.be/sHJSA3FgGd8.

[69] Chieh-Chun Chen, Navid Nikaein, and Alireza Mohammadi. "ECO-RAN: Multi-cell mobility management in an end-to-end 5G O-RAN network". In: *Mobile World Congress, 26-29 February 2024, Barcelona, Spain* (2024). URL: https://youtu.be/hlLt--WSQPc.

[70] Chieh-Chun Chen et al. "FlexApp: Flexible and Low-Latency xApp Framework for RAN Intelligent Controller". In: *ICC 2023 - IEEE International Conference on Communications*. 2023, pp. 5450–5456. DOI: 10.1109/ICC45041.2023.10278600.

[71] Chieh-Chun Chen et al. "xApp DevOps evolution and observable OAM in open RAN ecosystem". In: *Joint OSC/OSFG-OAI Workshop: End-to-End Reference Designs for O-RAN, 14-15 November 2023, Burlington, MA, USA* (2023). URL: https://www.eurecom.fr/publication/7863.

[72] Yongzhou Chen et al. "Channel-Aware 5G RAN Slicing with Customizable Schedulers". In: *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. Boston, MA: USENIX Association, Apr. 2023, pp. 1767–1782. ISBN: 978-1-939133-33-5. URL: https://www.usenix.org/conference/nsdi23/presentation/chen-yongzhou.

[73] Cilium. *Cilium BGP Control Plane (Beta)*. June 2024. URL: https://docs.cilium.io/en/stable/network/bgp-control-plane/.

[74] Luis M. Contreras, Ivan Bykov, and Krzysztof Grzegorz Szarkowicz. *5QI to DiffServ DSCP Mapping Example for Enforcement of 5G End-to-End Network Slice QoS*. Internet-Draft draft-cbs-teas-5qi-to-dscp-mapping-00. Work in Progress. Internet Engineering Task Force, Mar. 2024. 15 pp. URL: https://datatracker.ietf.org/doc/draft-cbs-teas-5qi-to-dscp-mapping/00/.

[75] O-RAN next Generation Research Group (nGRG) Contributed Research Report. *Digital Twin RAN: Key Enablers*. Tech. rep. Report ID: RR-2024-09. O-RAN Alliance, Oct. 2024.

[76] O-RAN next Generation Research Group (nGRG) Contributed Research Report. *Research Report on Digital Twin RAN Use Cases.* Tech. rep. Report ID: RR-2024-07. O-RAN Alliance, May 2024.

[77] Estefanía Coronado, Shah Nawaz Khan, and Roberto Riggio. “5G-EmPOWER: A Software-Defined Networking Platform for 5G Radio Access Networks”. In: *IEEE Transactions on Network and Service Management* 16.2 (2019), pp. 715–728. DOI: 10.1109/TNSM.2019.2908675.

[78] Jose Costa-Requena et al. “Software defined 5G mobile backhaul”. In: *1st International Conference on 5G for Ubiquitous Connectivity.* 2014, pp. 258–263. DOI: 10.4108/icst.5gu.2014.258054.

[79] Luca Deri et al. “nDPI: Open-source high-speed deep packet inspection”. In: *2014 International Wireless Communications and Mobile Computing Conference (IWCMC).* 2014, pp. 617–622. DOI: 10.1109/IWCMC.2014.6906427.

[80] Aaron Yi Ding and Marijn Janssen. “Opportunities for applications using 5G networks: requirements, challenges, and outlook”. In: *Proceedings of the Seventh International Conference on Telecommunications and Remote Sensing.* ICTRS ’18. New York, NY, USA: Association for Computing Machinery, 2018, 27–34. ISBN: 9781450365802. DOI: 10.1145/3278161.3278166. URL: https://doi.org/10.1145/3278161.3278166.

[81] Salvatore D’Oro et al. “dApps: Distributed Applications for Real-Time Inference and Control in O-RAN”. In: *IEEE Communications Magazine* 60.11 (2022), pp. 52–58. DOI: 10.1109/MCOM.002.2200079.

[82] Xenofon Foukas et al. “FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks”. In: *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies.* CoNEXT ’16. New York, NY, USA: Association for Computing Machinery, 2016, 427–441. ISBN: 9781450342926. DOI: 10.1145/2999572.2999599. URL: https://doi.org/10.1145/2999572.2999599.

[83] Xenofon Foukas et al. “Taking 5G RAN Analytics and Control to a New Level”. In: *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom ’23).* Madrid, Spain, 2023, pp. 1–16. ISBN: 9781450399906.

[84] HEAVY READING GABRIEL BROWN PRINCIPAL ANALYST. *TIP OpenRAN: Toward disaggregated mobile networking.* URL: https://telecominfraproject.com/openran/.

[85] Aditya Gudipati et al. “SoftRAN: Software defined radio access network”. In: *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking.* 2013, pp. 25–30.

[86] Tao Guo and Alberto Suárez. “Enabling 5G RAN Slicing With EDF Slice Scheduling”. In: *IEEE Transactions on Vehicular Technology* 68.3 (2019), pp. 2865–2877. DOI: 10.1109/TVT.2019.2894695.

[87] Scott D. Hahn et al. *Differentiated Services Quality of Service Policy Information Base.* RFC 3317. Mar. 2003. DOI: 10.17487/RFC3317. URL: https://www.rfc-editor.org/info/rfc3317.

[88] Yue Hu et al. *Where2comm: Communication-Efficient Collaborative Perception via Spatial Confidence Maps.* 2022. arXiv: 2209.12836 [cs.CV].

[89] Yong Xuan Huang, Kung Chun Wang, and Bi Shun Ke. "Accelerating 5G Service-Based Architecture with Ebpf". In: *Proceedings of the 2023 12th International Conference on Networks, Communication and Computing.* 2023, pp. 200–209.

[90] Mikel Irazabal and Navid Nikaein. "TC-RAN: A Programmable Traffic Control Service Model for 5G/6G SD-RAN". In: *IEEE Journal on Selected Areas in Communications* 42.2 (2024), pp. 406–419. DOI: 10.1109/JSAC.2023.3336162.

[91] Mikel Irazabal et al. "Dynamic Buffer Sizing and Pacing as Enablers of 5G Low-Latency Services". In: *IEEE Transactions on Mobile Computing* 21.3 (2022), pp. 926–939. DOI: 10.1109/TMC.2020.3017011.

[92] Muhammad Ali Jamshed et al. *Non-Terrestrial Networks for 6G: Integrated, Intelligent and Ubiquitous Connectivity.* 2024. arXiv: 2407.02184 [cs.NI]. URL: https://arxiv.org/abs/2407.02184.

[93] Tao Sun Jean Schwoerer Kevin Holley. *Network Architecture Evolution towards 6G v1.0.* URL: https://www.ngmn.org/publications/network-architecture-evolution-towards-6g.html.

[94] Baofeng Ji et al. "A survey of computational intelligence for 6G: Key technologies, applications and trends". In: *IEEE Transactions on Industrial Informatics* 17.10 (2021), pp. 7145–7154.

[95] David Johnson, Dustin Maas, and Jacobus Van Der Merwe. "NexRAN: Closed-loop RAN slicing in POWDER-A top-to-bottom open-source open-RAN use case". In: *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & CHaracterization.* 2022, pp. 17–23.

[96] Ahan Kak et al. *HexRAN: A Programmable Approach to Open RAN Base Station System Design.* 2024. arXiv: 2304.12560 [cs.NI]. URL: https://arxiv.org/abs/2304.12560.

[97] Ahan Kak et al. "ProSLICE: An Open RAN-based approach to Programmable RAN Slicing". In: *GLOBECOM 2022 - 2022 IEEE Global Communications Conference.* 2022, pp. 197–202. DOI: 10.1109/GLOBECOM48099.2022.10001497.

[98] Jaehong Kim et al. "OutRAN: co-optimizing for flow completion time in radio access network". In: *Proceedings of the 18th International Conference on emerging Networking EXperiments and Technologies.* 2022, pp. 369–385.

[99] Keith Kirkpatrick. "Software-defined networking". In: *Communications of the ACM* 56.9 (2013), pp. 16–19.

[100] Ravi Kokku et al. "NVS: A Substrate for Virtualizing Wireless Resources in Cellular Networks". In: *IEEE/ACM Transactions on Networking* 20.5 (2012), pp. 1333–1346. DOI: 10.1109/TNET.2011.2179063.

[101] Diego Kreutz et al. "Software-defined networking: A comprehensive survey". In: *Proceedings of the IEEE* 103.1 (2014), pp. 14–76.

[102] Adlen Ksentini and Navid Nikaein. "Toward Enforcing Network Slicing on RAN: Flexibility and Resources Abstraction". In: *IEEE Communications Magazine* 55.6 (2017), pp. 102–108. DOI: 10.1109/MCOM.2017.1601119.

[103] CZ.NIC Labs. *The BIRD Internet Routing Daemon.* Nov. 2020. URL: https://bird.network.cz/.

[104] Matti Latva-Aho, Kari Leppänen, et al. "Key drivers and research challenges for 6G ubiquitous wireless intelligence". In: (2019).

[105] Ta Dang Khoa Le and Navid Nikaein. "RAN Simulator Is NOT What You Need: O-RAN Reinforcement Learning for the Wireless Factory". In: *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking.* ACM MobiCom '23. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9781450399906. DOI: 10.1145/3570361.3615758. URL: https://doi.org/10.1145/3570361.3615758.

[106] Xhulio Limani et al. "Optimizing 5G Network Slicing: An End-to-End Approach with Isolation Principles". In: *2024 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN).* 2024, pp. 1–6. DOI: 10.1109/NFV-SDN61811.2024.10807481.

[107] Xingqin Lin et al. "6G Digital Twin Networks: From Theory to Practice". In: *IEEE Communications Magazine* 61.11 (2023), pp. 72–78. DOI: 10.1109/MCOM.001.2200830.

[108] Ruiqi Liu et al. *A Vision and An Evolutionary Framework for 6G: Scenarios, Capabilities and Enablers.* 2024. arXiv: 2305.13887 [cs.NI].

[109] Meng Lu et al. "C-ITS (cooperative intelligent transport systems) deployment in Europe: challenges and key findings". In: *25th ITS World Congress, Copenhagen, Denmark.* 2018, pp. 17–21.

[110] Robert MacDavid et al. "A P4-based 5G User Plane Function". In: *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR).* SOSR '21. New York, NY, USA: Association for Computing Machinery, 2021, 162–168. ISBN: 9781450390842. DOI: 10.1145/3482898.3483358. URL: https://doi.org/10.1145/3482898.3483358.

[111] Nick McKeown et al. "OpenFlow: enabling innovation in campus networks". In: *SIGCOMM Comput. Commun. Rev.* 38.2 (Mar. 2008), 69–74. ISSN: 0146-4833. DOI: 10.1145/1355734.1355746. URL: https://doi.org/10.1145/1355734.1355746.

[112] Usama Naseer et al. “Zero Downtime Release: Disruption-free Load Balancing of a Multi-Billion User Website”. In: *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*. SIGCOMM ’20. New York, NY, USA: Association for Computing Machinery, 2020, 529–541. ISBN: 9781450379557. DOI: 10.1145/3387514.3405885. URL: https://doi.org/10.1145/3387514.3405885.

[113] Kathleen Nichols et al. *Controlled Delay Active Queue Management*. RFC 8289. Jan. 2018. DOI: 10.17487/RFC8289. URL: https://www.rfc-editor.org/info/rfc8289.

[114] Navid Nikaein et al. “OpenAirInterface: A flexible platform for 5G research”. In: *ACM SIGCOMM Computer Communication Review* 44.5 (2014), pp. 33–38.

[115] Bruno Astuto A Nunes et al. “A survey of software-defined networking: Past, present, and future of programmable networks”. In: *IEEE Communications surveys & tutorials* 16.3 (2014), pp. 1617–1634.

[116] Joao Francisco Nunes Pinheiro et al. “5GECO: A Cross-domain Intelligent Neutral Host Architecture for 5G and Beyond”. In: *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2023, pp. 1–6. DOI: 10.1109/INFOCOMWKSHPS57453.2023.10225864.

[117] O-RAN ALLIANCE. *O-RAN Work Group 1 (Use Cases and Overall Architecture) O-RAN Architecture Description*. Tech. rep. Technical Specification O-RAN.WG1.OAD-R003-v12.00. June 2024.

[118] O-RAN ALLIANCE. *O-RAN Work Group 3 (Near-Real-time RAN Intelligent Controller and E2 Interface) Near-RT RIC APIs*. Tech. rep. Technical Specification O-RAN.WG3.RICAPI-R003-v02.00. June 2024.

[119] O-RAN Software Community. *RIC platform source code*. URL: https://gerrit.o-ran-sc.org/r/admin/repos/q/filter:ric-plt.

[120] O-RAN Software Community. *RMR Developer’s Guide*. URL: https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-lib-rmr/en/latest/developer-guide.html.

[121] O-RAN Software Community. *RMR vs NNG Sending Performance*. URL: https://wiki.o-ran-sc.org/display/RICP/RMR_nng_perf.

[122] O-RAN Software Community. *Suitability of gRPC for RMR Communications*. URL: https://wiki.o-ran-sc.org/display/RICP/gRPC+Evaluation.

[123] O-RAN Software Community. *xApp Writer’s Guide v2*. URL: https://wiki.o-ran-sc.org/display/RICP/Introduction+and+guides.

[124] Open Networking Foundation. *μONOS RIC architecture*. URL: https://docs.sd-ran.org/master/architecture.html.

[125] Imtiaz Parvez et al. “A Survey on Low Latency Towards 5G: RAN, Core Network and Caching Solutions”. In: *IEEE Communications Surveys & Tutorials* 20.4 (2018), pp. 3098–3130. DOI: 10.1109/COMST.2018.2841349.

[126] Girish Patel and Steven Dennett. "The 3GPP and 3GPP2 movements toward an all-IP mobile network". In: *IEEE Personal Communications* 7.4 (2000), pp. 62–64.

[127] Michele Polese et al. "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges". In: *IEEE Communications Surveys & Tutorials* 25.2 (2023), pp. 1376–1411.

[128] Matteo Pozza et al. "Network-In-a-Box: A Survey About On-Demand Flexible Networks". In: *IEEE Communications Surveys & Tutorials* 20.3 (2018), pp. 2407–2428. DOI: 10.1109/COMST.2018.2807125.

[129] Telecom Infra Project. *OpenRAN RIA at a Glance.* Feb. 2023. URL: https://telecominfraproject.com/openran/.

[130] 3GPP TSG SA Chair (Intel) Puneet Jain. *3GPP SA 6G Planning and Progress update.* URL: https://www.3gpp.org/ftp/Information/presentations/Presentations_2024.

[131] Yakov Rekhter, Susan Hares, and Tony Li. *A Border Gateway Protocol 4 (BGP-4).* RFC 4271. Jan. 2006. DOI: 10.17487/RFC4271. URL: https://www.rfc-editor.org/info/rfc4271.

[132] Mamoon M Saeed et al. "A comprehensive review on the users' identity privacy for 5G networks". In: *IET Communications* 16.5 (2022), pp. 384–399.

[133] Robert Schmidt, Chia-Yu Chang, and Navid Nikaein. "Slice Scheduling with QoS-Guarantee Towards 5G". In: *2019 IEEE Global Communications Conference (GLOBECOM).* 2019, pp. 1–7. DOI: 10.1109/GLOBECOM38437.2019.9013258.

[134] Robert Schmidt, Mikel Irazabal, and Navid Nikaein. "FlexRIC: an SDK for next-generation SD-RANs". In: *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies.* CoNEXT '21. New York, NY, USA: Association for Computing Machinery, 2021, 411–425. ISBN: 9781450390989. DOI: 10.1145/3485983.3494870. URL: https://doi.org/10.1145/3485983.3494870.

[135] Robert Schmidt and Navid Nikaein. "RAN Engine: Service-Oriented RAN Through Containerized Micro-Services". In: *IEEE Transactions on Network and Service Management* 18.1 (2021), pp. 469–481. DOI: 10.1109/TNSM.2021.3057642.

[136] Susanna Schwarzmann et al. "An Intelligent User Plane to Support In-Network Computing in 6G Networks". In: *ICC 2023-IEEE International Conference on Communications.* IEEE. 2023, pp. 1100–1105.

[137] Conor Sexton et al. "5G: Adaptable networks enabled by versatile radio access technologies". In: *IEEE Communications Surveys & Tutorials* 19.2 (2017), pp. 688–720.

[138] Pankaj Sharma. "Evolution of mobile wireless communication networks-1G to 5G as well as future prospective of next generation communication network". In: *International Journal of Computer Science and Mobile Computing* 2.8 (2013), pp. 47–53.

[139] Zhaogang Shu and Tarik Taleb. "A Novel QoS Framework for Network Slicing in 5G and Beyond Networks Based on SDN and NFV". In: *IEEE Network* 34.3 (2020), pp. 256–263. DOI: 10.1109/MNET.001.1900423.

[140] Tigera. *Configure BGP peering*. June 2024. URL: http://ccrma.stanford.edu/~jos/bayes/bayes.html.

[141] Juha-Matti Tilli and Raimo Kantola. "Data plane protocols and fragmentation for 5G". In: *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*. 2017, pp. 207–213. DOI: 10.1109/CSCN.2017.8088623.

[142] Maria Tsampazi et al. "PandORA: Automated Design and Comprehensive Evaluation of Deep Reinforcement Learning Agents for Open RAN". In: *IEEE Transactions on Mobile Computing* (2024).

[143] Nokia Thierry Van de Velde. *IPv6 in 5G*. URL: https://www.ipv6council.be/?p=597.

[144] Marcos A. M. Vieira et al. "Fast Packet Processing with eBPF and XDP: Concepts, Code, Challenges, and Applications". In: *ACM Comput. Surv.* 53.1 (Feb. 2020). ISSN: 0360-0300. DOI: 10.1145/3371038. URL: https://doi.org/10.1145/3371038.

[145] Xuyu Wang, Shiwen Mao, and Michelle X Gong. "An overview of 3GPP cellular vehicle-to-everything standards". In: *GetMobile: Mobile Computing and Communications* 21.3 (2017), pp. 19–25.

[146] Jeroen Wigard et al. "Ubiquitous 6G Service Through Non-Terrestrial Networks". In: *IEEE Wireless Communications* 30.6 (2023), pp. 12–18. DOI: 10.1109/MWC.001.2300173.

[147] Dariusz Wypiór, Mirosław Klinkowski, and Igor Michalski. "Open RAN—Radio Access Network Evolution, Benefits and Market Trends". In: *Applied Sciences* 12.1 (2022). ISSN: 2076-3417. DOI: 10.3390/app12010408. URL: https://www.mdpi.com/2076-3417/12/1/408.

[148] Xiaohu You et al. "Towards 6G wireless communication networks: Vision, enabling technologies, and new paradigm shifts". In: *Science China Information Sciences* 64 (2021), pp. 1–74.

[149] Zhi-Li Zhang et al. "Towards a Software-Defined, Fine-Grained QoS Framework for 5G and Beyond Networks". In: *Proceedings of the ACM SIGCOMM 2021 Workshop on Network-Application Integration*. NAI'21. New York, NY, USA: Association for Computing Machinery, 2021, 7–13. ISBN: 9781450386333. DOI: 10.1145/3472727.3472798. URL: https://doi.org/10.1145/3472727.3472798.