ALMA MATER STUDIORUM – UNIVERSITY OF BOLOGNA

SCHOOL OF ENGINEERING

Department of
Computer Science – Science and Engineering
DISI

Master's Degree in Computer Engineering

Master's Thesis
in
Mobile Systems

Enabling O-RAN Closed-Loop Control over Wi-Fi-Based Untrusted Non-3GPP Access Networks

Candidate:

Zakaria Traka

Advisor:

Prof. Paolo Bellavista

Coadvisor:

Prof. Armir Bujari
Phd Angelo Feraudo
Phd Alessandro Calvio

Academic Year
2024–2025

Contents

List of Figures

List of Tables

List of Abbreviations

3GPP	3rd Generation Partnership Project
5GC	5G Core
5GS	5G System
AAA	Authentication, Authorization and Accounting
AF	Application Function
AMF	Access and Mobility Management Function
AP	Access Point
APER	Aligned Packed Encoding Rules
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
ATSSS	Access Traffic Steering, Switching and Splitting
AUSF	Authentication Server Function
BAR	Buffering Action Rule
CGI	Cell Global Identifier
CN	Core Network
CQI	Channel Quality Indicator
CU	Central Unit
CUPS	Control and User Plane Separation
DHCP	Dynamic Host Configuration Protocol
DL	Downlink
DNN	Data Network Name
DPD	Dead Peer Detection
DRB	Data Radio Bearer
DU	Distributed Unit
E2	O-RAN E2 Interface
E2AP	E2 Application Protocol
E2SM	E2 Service Model
E2SM-KPM	Key Performance Measurements Service Model
E2SM-RC	RAN Control Service Model
EAP	Extensible Authentication Protocol
eMBB	enhanced Mobile Broadband

ESP	Encapsulating Security Payload
FAR	Forwarding Action Rule
gNB	gNodeB
GRE	Generic Routing Encapsulation
GRO	Generic Receive Offload
GSO	Generic Segmentation Offload
GTP-U	GPRS Tunnelling Protocol User Plane
GUTI	Globally Unique Temporary Identifier
HO	Handover
HTTP	Hypertext Transfer Protocol
ICV	Integrity Check Value
IEEE	Institute of Electrical and Electronics Engineers
IKEv2	Internet Key Exchange version 2
IP	Internet Protocol
IPsec	IP Security
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
KPM	Key Performance Measurements
LADN	Local Area Data Network
MAC	Medium Access Control
MCC	Mobile Country Code
MCS	Modulation and Coding Scheme
MEC	Multi-access Edge Computing
ML	Machine Learning
mMTC	massive Machine-Type Communications
MNC	Mobile Network Code
MOBIKE	Mobility and Multihoming Protocol for IKEv2
MPDU	MAC Protocol Data Unit
N2	Interface between (R)AN and AMF
N3	Interface between (R)AN and UPF
N3IWF	Non-3GPP Interworking Function
NAS	Non-Access Stratum
NAT	Network Address Translation
NDS	Network Domain Security
Near-RT RIC	Near Real-Time RAN Intelligent Controller
NEF	Network Exposure Function
NF	Network Function
NGAP	Next Generation Application Protocol
NR	New Radio
NR CGI	NR Cell Global Identifier
NRF	Network Repository Function

NRT RIC	Non-Real Time RIC
NSSAI	Network Slice Selection Assistance Information
NSSF	Network Slice Selection Function
O-RAN	Open Radio Access Network
OER	Octet Encoding Rules
PCF	Policy Control Function
PDCP	Packet Data Convergence Protocol
PDR	Packet Detection Rule
PDU	Protocol Data Unit
PFCP	Packet Forwarding Control Protocol
PLMN	Public Land Mobile Network
PRB	Physical Resource Block
PSA	PDU Session Anchor
QER	QoS Enforcement Rule
QFI	QoS Flow Identifier
QoS	Quality of Service
RAN	Radio Access Network
RAT	Radio Access Technology
RC	RAN Control
REST	Representational State Transfer
RIC	RAN Intelligent Controller
RLC	Radio Link Control
RMR	RIC Message Router
RRC	Radio Resource Control
RRU	Remote Radio Unit
SA	Security Association
SBA	Service-Based Architecture
SBI	Service-Based Interface
SCTP	Stream Control Transmission Protocol
SDAP	Service Data Adaptation Protocol
SDL	Shared Data Layer
SMF	Session Management Function
SMO	Service Management and Orchestration
SPI	Security Parameters Index
SSC	Session and Service Continuity
SSID	Service Set Identifier
TCP	Transmission Control Protocol
TEID	Tunnel Endpoint Identifier
TLS	Transport Layer Security
TNGF	Trusted Non-3GPP Gateway Function
TSO	TCP Segmentation Offload

UDP	User Datagram Protocol
UDM	Unified Data Management
UDR	Unified Data Repository
UE	User Equipment
UL	Uplink
UPF	User Plane Function
URLLC	Ultra-Reliable Low-Latency Communications
URR	Usage Reporting Rule
VPN	Virtual Private Network
WLAN	Wireless Local Area Network
WPA2	Wi-Fi Protected Access 2
XFRM	Linux IPsec Transform Framework

Abstract

Fifth-generation (5G) networks introduce a Service-Based Architecture that, for the first time within a unified core architecture, grants non-3GPP access technologies, such as Wi-Fi, satellite, and fixed broadband, access to the 5G Core alongside traditional cellular radio. The Non-3GPP Interworking Function (N3IWF) is the 5G Core gateway that connects UEs accessing via untrusted non-3GPP networks to the core network over IPsec tunnels, allowing them to access the same services as cellular terminals.

The O-RAN standard defines near-real-time control of RAN nodes through the RAN Intelligent Controller (Near-RT RIC) and the E2 interface, enabling portable, vendor-neutral radio resource management (RRM) applications. Extending this framework to non-3GPP access would allow operators to manage and optimize all access technologies through a single control plane, enabling cross-technology decisions such as load balancing between cellular and Wi-Fi or coordinated mobility across heterogeneous links. However, only 3GPP RAN elements (gNB, O-CU, O-DU) are currently defined as E2 Nodes: non-3GPP access falls entirely outside the O-RAN perimeter, preventing this unified approach.

This thesis bridges the gap by integrating the N3IWF as an O-RAN-compliant E2 Node, focusing on Wi-Fi as the target non-3GPP radio technology, since it represents the most common untrusted access scenario in current 5G deployments. A co-located agent implements the E2 Application Protocol and exposes two RAN Functions (E2SM-KPM and E2SM-RC) to the Near-RT RIC. Through E2SM-KPM, Wi-Fi station-level metrics are mapped onto 3GPP-aligned measurement definitions, enabling xApps to monitor non-3GPP and 3GPP access nodes with uniform logic. Through E2SM-RC, the Near-RT RIC can issue control actions to the gateway; specifically, an N2-based handover mechanism between N3IWF instances is implemented, with a Handover xApp closing the control loop.

The N2 handover between N3IWF instances represents a significant challenge: unlike classical 3GPP mobility, there is no radio-level RRM layer, no Xn interface between access nodes, and data-plane continuity depends on

IPsec tunnels rather than radio bearers. The proposed solution synchronizes the full IKE/IPsec Security Association state between gateways through the N2 handover signalling, leverages the MOBIKE protocol for tunnel re-binding after the Wi-Fi switch, and introduces an early path switch at the UPF combined with target-side buffering to mitigate packet loss during the transition.

Experimental evaluation on a four-node physical testbed demonstrates a mean handover latency of 815 ms, primarily limited by the physical Wi-Fi re-association process. Target-side buffering reduces peak UDP downlink packet loss from 54% to 0.4% at 60 Mbps. The results also confirm that IPsec state synchronization is necessary for stable operation under repeated handovers: disabling it approximately doubles the latency (from ∼815 ms to ∼1 961 ms) and causes progressive AMF failures. Overall, the work shows that O-RAN closed-loop control can be extended to non-3GPP access nodes, bringing Wi-Fi gateways under the same management framework used for 3GPP RAN elements.

Keywords: O-RAN; E2 Interface; Near-RT RIC; N3IWF; non-3GPP access; E2SM-KPM; E2SM-RC; N2 Handover; IPsec; MOBIKE; free5GC;

Chapter 1

Introduction

5G networks adopt a Service-Based Architecture (SBA) that, for the first time within a unified core architecture, gives non-3GPP access technologies entry to the 5G Core alongside traditional cellular radio [2]. Non-3GPP access refers to any access technology not specified by 3GPP, such as Wi-Fi, satellite links, or fixed broadband. For untrusted non-3GPP access, the 5G Core defines the Non-3GPP Interworking Function (N3IWF), a gateway that terminates the N2 (control-plane) and N3 (user-plane) reference points toward the core network and connects UEs via IKEv2/IPsec tunnels [2, 6]. Through the N3IWF, a UE attached to a Wi-Fi access point can authenticate, establish PDU Sessions, and benefit from equivalent mobility and session continuity services.

The O-RAN standard builds on the 5G foundation by introducing a programmable near-real-time control layer centered on the Near Real-Time RAN Intelligent Controller (Near-RT RIC) [7, 3]. By connecting to RAN elements via the E2 interface [8], the Near-RT RIC hosts *xApps* that execute data-driven optimization within a 10 ms–1 s control loop. Through the E2SM-KPM and E2SM-RC service models [4, 5], 3GPP-compliant nodes such as gNBs and O-CU/O-DU components expose standardized telemetry and accept control actions, effectively externalizing RAN optimization logic to the Near-RT RIC. Extending this framework to non-3GPP access would allow operators to manage all access technologies through a single control plane, enabling cross-technology decisions such as load balancing between cellular and Wi-Fi or coordinated mobility across heterogeneous links.

However, only 3GPP RAN elements are currently defined as E2 Nodes in the O-RAN specifications; non-3GPP access nodes do not expose RAN Functions to the Near-RT RIC [3]. The telemetry-driven optimization available for gNBs does not extend to the Wi-Fi segment. This gap in observability and control prevents operators from applying a uniform closed-loop manage-

ment approach across heterogeneous access technologies. The N3IWF sits at the center of this gap: despite being the 5G Core component that integrates untrusted Wi-Fi access, it has no E2AP implementation, no RAN Function definitions, and no mechanism for reporting Wi-Fi performance data through standardized O-RAN service models.

This thesis addresses this gap by enabling the N3IWF to operate as an O-RAN-compliant E2 Node. A co-located E2 agent implements the E2 Application Protocol and exposes RAN Functions to the Near-RT RIC. Wi-Fi station-level metrics are mapped onto E2SM-KPM measurement definitions, allowing xApps to apply the same monitoring logic to both non-3GPP and 3GPP access nodes. Control actions are supported via E2SM-RC, enabling the Near-RT RIC to trigger procedures on the N3IWF through the standard E2 control path. By abstracting Wi-Fi access behind O-RAN service models, the approach decouples xApp control logic from gateway-specific internals. Figure 1.1 provides a high-level overview of this architecture.

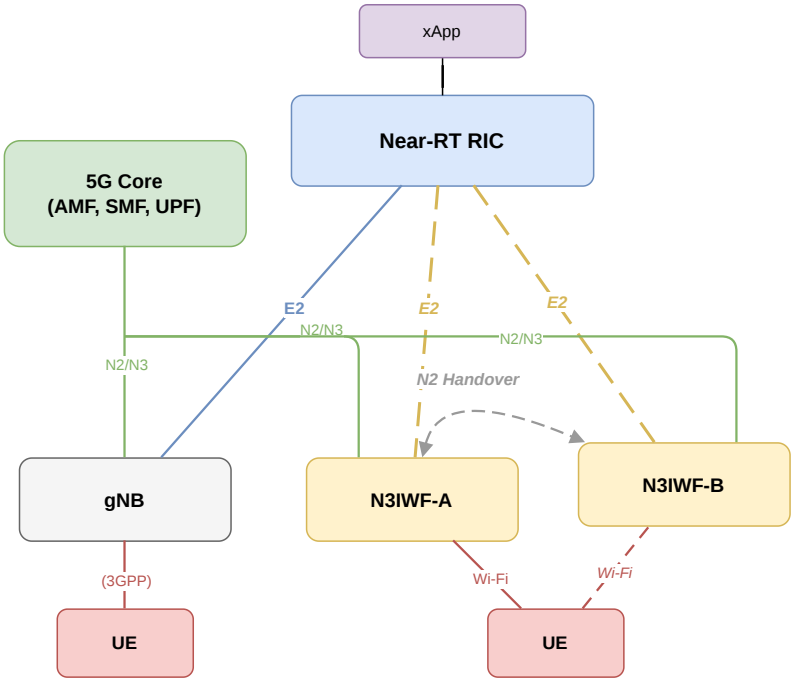

Figure 1.1: High-level system overview. The Near-RT RIC controls 3GPP nodes (gNB) via the standard E2 interface. This work extends E2 connectivity to N3IWF instances (dashed lines), enabling RIC-driven monitoring and control of non-3GPP access, including N2-based handover between N3IWF gateways.

As a concrete use case of the E2SM-RC interface, an N2-based handover mechanism between N3IWF instances is implemented. A Handover xApp subscribes to KPM reports from all E2 Nodes, evaluates UE distribution, and issues control requests to rebalance load across gateways. Adapting the standard 3GPP N2 handover procedure to untrusted non-3GPP access introduces additional challenges: unlike cellular RAN handover, session continuity depends on IPsec tunnel state rather than radio bearer context. The proposed mechanism therefore includes IPsec Security Association synchronization between source and target N3IWFs, MOBIKE-based tunnel re-binding after Wi-Fi reassociation, and target-side buffering to preserve packet continuity during the transition. The implementation and evaluation of these adaptations demonstrate that the O-RAN near-real-time control loop can be extended beyond 3GPP RAN elements to non-3GPP access domains. This work focuses on untrusted non-3GPP access via the N3IWF within a single

5G Core domain and does not address trusted non-3GPP access or inter-core mobility scenarios.

Contributions The main contributions of this thesis are:

1. The design and implementation of an E2 Node integration for the N3IWF, including a mapping of Wi-Fi performance metrics to standardized E2SM-KPM definitions and support for E2SM-RC control, enabling O-RAN-compliant monitoring and control of non-3GPP access.

2. An N2-based inter-N3IWF handover mechanism that ensures session continuity in an IPsec-based access domain through Security Association synchronization, MOBIKE-assisted execution, early path switching, and target-side buffering.

3. An experimental evaluation on a four-node testbed demonstrating a mean handover latency of 815 ms (dominated by the physical Wi-Fi reassociation phase), peak UDP downlink packet loss reduced from 54% to 0.4% at 60 Mbps through target-side buffering, and stable operation under repeated mobility events.

Thesis Structure The remainder of this thesis is organized as follows. Chapter 2 provides the technical background on 5G Core architecture, non-3GPP access via N3IWF, IPsec/IKEv2, the N2 handover procedure, and the O-RAN Near-RT RIC with its E2 interface and service models. Chapter 3 reviews related work and positions the contributions of this thesis with respect to the existing literature. Chapter 4 presents the system design, focusing first on the E2 Node integration and then on the N2 handover adaptation as a control use case. Chapter 5 details the implementation of the co-located E2 agent, the Handover xApp, the N3IWF handover logic, and the required core network modifications. Chapter 6 describes the experimental setup and presents the evaluation results. Chapter 7 concludes with a summary of contributions, limitations, and future directions.

Chapter 2

Background

This chapter provides the technical foundations required to understand the contributions of this thesis. It covers the evolution towards 5G and the core architecture, non-3GPP access integration through the N3IWF, the IPsec/IKEv2 protocols underpinning secure non-3GPP tunnels, the O-RAN architecture with its E2 interface and service models, and the N2-based handover procedure.

2.1 Mobile Networks and 5G

Mobile communications progressed from analog voice (1G) through digital systems with data capabilities (2G/3G) to the all-IP, OFDMA-based 4G LTE architecture that enabled broadband mobile internet [9, 10, 11]. 5G, standardized by 3GPP from Release 15 onward, goes beyond raw throughput improvements and defines three broad service categories [12]:

- **eMBB** (enhanced Mobile Broadband): peak rates up to 20 Gbps, supporting 4K/8K video, AR/VR applications;
- **URLLC** (Ultra-Reliable Low-Latency Communications): latency below 1 ms with 99.999% reliability, targeting industrial automation and autonomous vehicles;
- **mMTC** (massive Machine-Type Communications): support for up to 1 million devices per km^2, enabling large-scale IoT deployments.

The 5G ecosystem is shaped by several standardization bodies. The **3rd Generation Partnership Project (3GPP)** [13] defines the 5G system:

Release 15 (2018) introduced 5G NR and the 5G Core (5GC), the first complete 5G system specification. 3GPP organizes its work into Technical Specification Groups covering RAN, Service & System Aspects (SA), and Core Network & Terminals (CT) [14, 10]. Complementing 3GPP, the **O-RAN Alliance** [7] promotes a disaggregated and vendor-interoperable RAN architecture with open interfaces (covered in Section 2.4). The **ITU** sets overarching performance targets through its IMT-2020 program [12], and the **IEEE** maintains the 802.11 Wi-Fi standards [15], directly relevant to the non-3GPP access scenario addressed in this thesis.

2.2 5G Core Architecture

The **5G Core (5GC)** departs from the monolithic network elements of the 4G Evolved Packet Core (EPC) by adopting a **Service-Based Architecture (SBA)** [2]. In SBA, each **Network Function (NF)** is a software component that exposes its services through well-defined APIs and can be independently developed, deployed, and scaled. NFs act as service producers and consumers, discovering each other dynamically through the NRF (Network Repository Function). This modular approach allows operators to mix implementations from different vendors and to deploy NFs as containerized microservices.

Figures 2.1 and 2.2 illustrate the 5GS architecture in the service-based and reference-point representations, respectively. NFs interact through two distinct communication paradigms [2]:

- **Service-Based Interface (SBI)**: Used for communication *among* 5GC NFs. The SBI uses HTTP/2 over TCP/TLS, with services defined using OpenAPI specifications. NFs register their services with the NRF (Network Repository Function) and discover other services dynamically. The SBI is represented in architecture diagrams as connections to a common "bus" (e.g., Namf, Nsmf, Nausf for AMF, SMF, AUSF services respectively).

- **Reference Points**: Used for communication between the 5GC and external entities (UE, NG-RAN, Data Networks) or for specific control/user plane protocols. These use point-to-point interfaces with dedicated protocols (e.g., NGAP over SCTP for N2, GTP-U over UDP for N3, PFCP for N4).

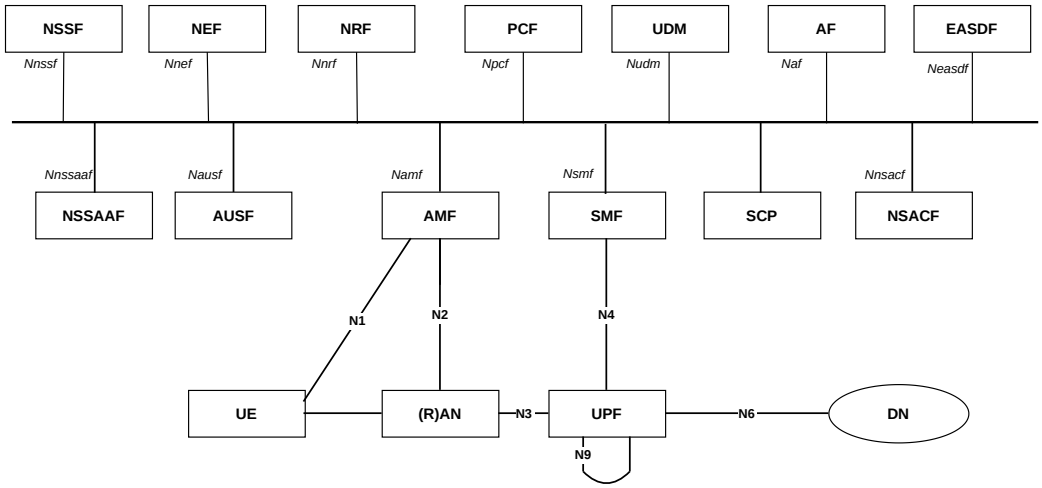

Figure 2.1: 5G System architecture – service-based representation (based on TS 23.501, Fig. 4.2.3-1).

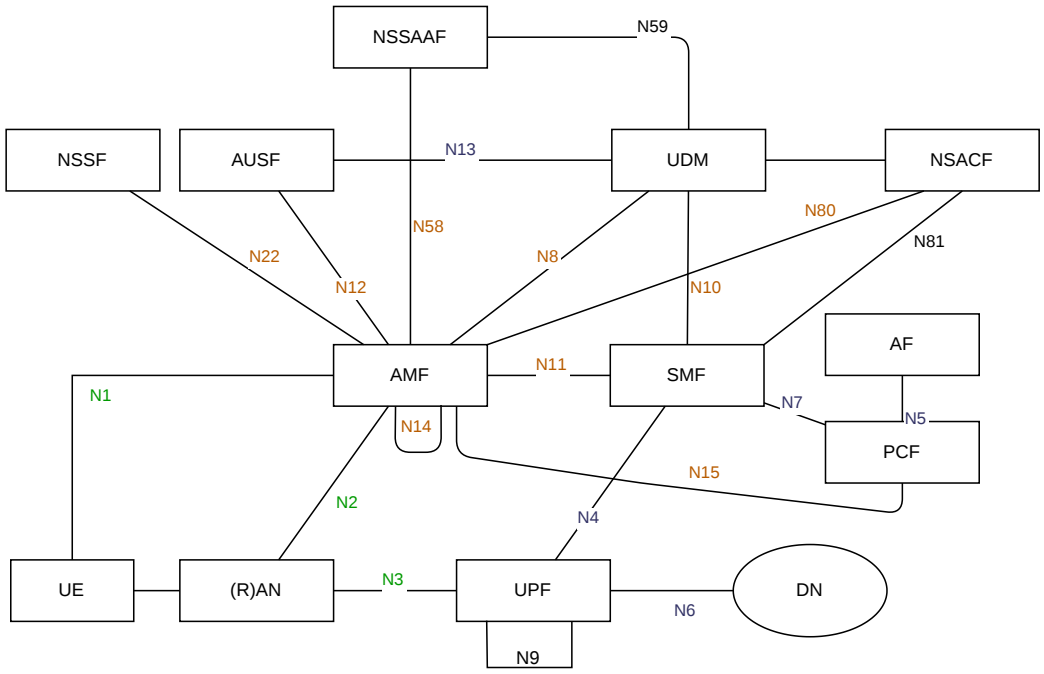

Figure 2.2: 5G System architecture – reference point representation (based on TS 23.501, Fig. 4.2.3-2).

Open-source 5GC implementations Open-source projects such as free5GC [16] (Go, NYCU) and Open5GS (C) provide full Release 15+ NF implementations deployable as standalone processes or containers, enabling research experimentation with the complete 5GC stack.

2.2.1 Reference Points and Interface Realization

The 5G System defines *reference points* (e.g., N1, N2, N3) and *service-based interfaces* (e.g., Namf, Nsmf) to describe interactions between network functions and access nodes [2]. Many NF-to-NF reference points (e.g., N11, N12, N8) are realized by the Service Based Architecture (SBA), i.e., HTTP/2 over TCP with TLS and JSON payload encoding [2, 17].

Communication paradigms: SBA vs. E2 A relevant aspect in multi-domain integration is the coexistence of two fundamentally different communication paradigms. Within the 5GC, the Service-Based Architecture relies on web-like interfaces (HTTP/2 over TLS with JSON/OpenAPI payloads) for NF-to-NF interactions. In O-RAN, the E2 interface relies on ASN.1-defined messages encoded with APER and transported over SCTP. This mismatch between an API-centric control plane and a binary-optimized signaling stack becomes relevant when integrating O-RAN control logic with 5GC-based functions [2, 17, 18, 8].

Core reference points used in this thesis

Table 2.1 summarizes the most relevant reference points for access, user plane forwarding, and the key control-plane interactions.

Among NF-to-NF control-plane interfaces, the most relevant to this work is **N11** (AMF–SMF), over which the AMF invokes `Nsmf_PDUSession_UpdateSMContext` during handover to coordinate path switching with the SMF [2, 17]. Other SBI-based reference points (e.g., N12 for authentication, N7/N15 for policy) follow the same HTTP/2 pattern but are not directly involved in the handover procedure.

2.2.2 Network Functions Overview

Table 2.2 summarizes the NFs defined by TS 23.501 [2]. The four most relevant to this thesis (N3IWF, AMF, SMF, and UPF) are detailed in the following subsections.

Table 2.1: Core 5GS reference points: endpoints and protocol realization

Ref.	**End-points**	**Plane**	**Protocol / realization (typical)**
N1	UE ↔ AMF	Access CP (NAS)	NAS (logical UE–AMF interface). Over 3GPP access, NAS is transported via NG-RAN (e.g., NAS Transport in NGAP). Over untrusted non-3GPP, NAS is carried inside secure tunnels via N3IWF [2, 19, 1].
N2 (NG-C)	(R)AN ↔ AMF	Access CP	NGAP over SCTP/IP. Supports UE context procedures, NAS transport, and paging (for NG-RAN). For N3IWF, NGAP still applies, but paging over non-3GPP access is not supported [20, 19, 2].
N3 (NG-U)	(R)AN ↔ UPF	Access UP	GTPv1-U over UDP/IP (default UDP port 2152). Used for user-plane tunneling both for gNB–UPF and N3IWF–UPF [21, 2].
N4	SMF ↔ UPF	5GC CP (control of UP)	PFCP over UDP/IP (PFCP destination port 8805 for requests). SMF installs PDR/FAR/QER/URR/BAR rules in the UPF [22, 2].
N6	UPF ↔ DN	External connectivity	IP/Ethernet interconnect toward Data Networks. Traffic is forwarded according to SMF-programmed rules; NAT may be deployed depending on operator design [2].
N9	UPF ↔ UPF	5GC user plane	GTPv1-U over UDP/IP for inter-UPF chaining (e.g., PSA + ULCL/BP topologies) [21, 2].
NWu	UE ↔ N3IWF	Non-3GPP access	Secure tunneling over untrusted non-3GPP access. User plane uses IPsec Child SAs and GRE; QFI is mapped to Child SAs for UL/DL traffic handling [2, 1].

Table 2.2: 5G Core Network Functions Overview

NF	**Primary Responsibilities**
AMF	NAS signaling termination, registration, connection and mobility management, security context handling
SMF	PDU session management (establishment, modification, release), UPF selection and control, IP address allocation
UPF	Packet routing and forwarding, QoS enforcement, traffic measurement, GTP-U tunneling
NRF	NF registration and discovery, service authorization, NF profile management
AUSF	Authentication procedures (5G-AKA, EAP-AKA'), authentication vector generation
UDM	Subscription data management, authentication credential processing, user identification
UDR	Storage of subscription data, policy data, and application data accessed by UDM, PCF, NEF
PCF	Policy rules provision (access, session, QoS), charging control, policy decision
NSSF	Slice selection assistance, AMF selection for slice, S-NSSAI determination
NEF	Secure exposure of network capabilities to external applications (AFs), API translation
N3IWF	Untrusted non-3GPP access integration, IPsec tunnel termination, NAS relay

Control Plane vs. User Plane NFs Following the Control and User Plane Separation (CUPS) principle, the 5GC maintains a clear separation:

- **Control Plane NFs**: AMF, SMF, NRF, AUSF, UDM, UDR, PCF, NSSF, NEF handle signaling, policy, and session management. They communicate primarily via the SBI.
- **User Plane NF**: The UPF is the only NF in the user plane, responsible for forwarding data packets between the RAN and external data networks.

The following subsections detail the AMF, SMF, and UPF, as their interactions are central to the handover and session continuity mechanisms addressed in this work.

2.2.3 Access and Mobility Management Function (AMF)

Every UE interaction with the 5GC passes through the **Access and Mobility Management Function (AMF)**, which terminates both the **N1** (NAS signaling) and **N2** (access-network control-plane) interfaces [2]. Its core responsibilities include:

- **NAS termination and security**: termination of N1 (UE–AMF) and N2 ((R)AN–AMF), including NAS ciphering and integrity protection. The AMF also acts as the **SEAF** (Security Anchor Functionality) in the 5G security architecture [2, 23].
- **Registration and connection management**: management of the UE registration state, tracking-area updates, and NAS signaling connection state (CM-IDLE vs CM-CONNECTED) [2].
- **Mobility management**: coordination of UE mobility events, including N2-based handover and inter-AMF context transfer [2, 1].
- **SM signaling transport**: transparent relay of Session Management (SM) messages between UE and SMF over N11 [2, 1].

Non-3GPP access considerations When the UE reaches the 5GC through an untrusted non-3GPP network, the AMF role is unchanged: it still terminates N1 and N2, but the access-side endpoint becomes the **N3IWF** instead of a gNB. The N3IWF relays NAS signaling on behalf of the UE, and the AMF maintains a separate access-specific context when the UE is

concurrently served over 3GPP and non-3GPP access [2, 1, 23]. A key behavioral difference is **reachability**: the UE cannot be paged over non-3GPP access, so paging-dependent reachability procedures are not applicable [2, 1].

2.2.4 Session Management Function (SMF)

While the AMF owns the access and mobility context, the **Session Management Function (SMF)** owns the *PDU Session* lifecycle. Once the AMF selects an SMF for a given session, that SMF remains bound to it for all subsequent session management procedures [2]. Its core functions are:

- **PDU Session lifecycle**: establishment, modification, and release of PDU Sessions, including UE IP address allocation and maintaining tunnel connectivity between the UPF and the access node [2].
- **UPF selection and control**: selects one or more UPFs for a PDU Session and programs their forwarding rules over **N4** (PFCP), including PDR/FAR updates during mobility events [2].
- **N2 SM information**: generates access-network-specific session parameters (e.g., GTP-U tunnel endpoints, QoS profiles) transported by the AMF over N2 to the access node [2].
- **SSC mode enforcement**: determines and enforces the Session and Service Continuity (SSC) mode, which governs whether the UPF anchor is preserved during mobility [2].

Non-3GPP access considerations When the access node is an N3IWF rather than a gNB, the SMF role is unchanged: it still provisions QoS and tunnel parameters via N2 SM information relayed by the AMF. The difference is that the N3IWF maps QoS flows to IPsec Child SAs rather than radio bearers [2, 1, 23]. Paging is not available over non-3GPP access, so SMF-triggered downlink activation follows a different reachability model [2, 1].

2.2.5 User Plane Function (UPF)

The **User Plane Function (UPF)** is the sole data-plane element in the 5GC. It sits between the access network (N3) and external Data Networks (N6), and can be chained with other UPFs over N9. The SMF programs its behavior over N4 using PFCP [2, 22]. In effect, the SMF decides *what* to do with packets, and the UPF executes it at line rate. Its key capabilities include:

- **Packet routing and forwarding**: routes packets between the access node (via GTP-U on N3) and external Data Networks (N6), applying forwarding actions programmed by the SMF [2].
- **Mobility anchoring**: acts as the PDU Session Anchor (PSA), providing IP continuity when the UE moves between access nodes. During handover, the SMF updates the UPF's forwarding rules to redirect traffic to the target access node [2].
- **Downlink buffering and End Markers**: buffers downlink packets when the UE is unreachable and sends End Marker packets on the old N3 tunnel to signal path switching during mobility [2].
- **QoS enforcement**: enforces uplink/downlink rate limits and applies transport-level marking as instructed by the SMF [2].

PFCP rule model On N4, the SMF programs the UPF through PFCP [22]. Each PDU Session maps to a PFCP session containing: **PDRs** (packet detection rules) that classify incoming traffic, **FARs** (forwarding action rules) that specify the forwarding destination (e.g., the GTP-U tunnel endpoint of the serving access node), **QERs** (QoS enforcement rules), and optionally **BARs** (buffering action rules). During handover, the SMF issues a PFCP Session Modification to update the DL FAR with the target access node's N3 tunnel information, effecting the downlink path switch.

Non-3GPP access considerations From the UPF perspective, serving an N3IWF is no different from serving a gNB: N3 still carries GTP-U. The difference is on the access side, where the N3IWF maps QoS flows to IPsec Child SAs rather than radio bearers [1, 21]. QoS flows are carried over **GRE** between UE and N3IWF, with the **QFI** encoded in the GRE header [1]. Downlink buffering at the UPF is still possible, but **paging is not supported over non-3GPP access**, so paging-driven reachability procedures are not applicable in N3IWF-only scenarios [2, 1].

2.2.6 Handover in 5G

Handover (HO) is the procedure by which a UE's serving access node is changed while maintaining active sessions. In 5G NR, two main handover types are defined for connected-mode UEs [24, 1]:

- **Xn-based handover**: The source and target RAN nodes communicate directly over the Xn interface, transferring UE context without

core involvement in every step. This is the preferred path when an Xn link exists.

- **N2-based handover** (also called NG handover): When no direct Xn interface is available between source and target, the handover is routed through the 5G Core via the N2 (NGAP) interface. The AMF coordinates the procedure, the SMF updates the user-plane path, and the UPF switches the GTP-U tunnel to the target.

A third type, **F1-based handover**, applies to disaggregated gNB architectures (O-CU/O-DU) where mobility occurs between DUs managed by the same CU, but is not relevant to this thesis.

In all cases, the goal is to preserve ongoing PDU Sessions and minimize service interruption. For non-3GPP access via N3IWF, no Xn-equivalent interface exists between N3IWF instances, making the N2-based handover the only viable core-coordinated option. The following subsections describe the N2 handover procedure in detail.

N2-based Handover Procedure

The N2-based handover (also called NG interface handover) is a 5G handover procedure coordinated via the 5G Core (5GC) using the N2 interface (gNB–AMF). In N2 handover, there is no direct Xn link between source and target RAN nodes, so the source gNB forwards handover requests through the AMF to the target gNB. In contrast, an Xn-based handover uses the direct Xn interface between gNBs and is handled largely within the RAN (minimal core involvement). In practice, N2 handover is used when the target gNB is not Xn-connected to the source, whereas Xn handover is used when an Xn interface exists. Both procedures aim to maintain ongoing PDU Sessions (via SMF/UPF) and UE context, but N2 handover involves more core-network signaling (N2, N11, N4), while Xn handover proceeds with local RAN signaling.

Preparation Phase of N2 Handover

The preparation phase of the Inter NG-RAN node N2-based handover is specified in 3GPP TS 23.502 (clause 4.9.1.3.2) [1]. The procedure involves the Source RAN (S-RAN), the AMF, the Target RAN (T-RAN), the relevant SMF(s), and UPF(s). Figure 2.3 and the description below present a simplified version of the preparation phase, focusing on the core message flow. Conditional steps such as AMF relocation (T-AMF selection and

`CreateUEContext` exchange), error handling, and secondary RAT reporting are omitted; the complete procedure is detailed in 3GPP TS 23.502 [1].

1. **S-RAN → AMF: Handover Required.** The S-RAN decides that a handover is needed (based on measurement reports, load conditions, or other criteria), selects the Target ID, and sends an NGAP *Handover Required* message to the AMF. The message carries the Target ID, the Source-to-Target Transparent Container (RRC context, UE capabilities), the Handover Type, and the list of PDU Sessions with N2 SM information [1, 19].

2. **AMF → SMF: `Nsmf_PDUSession_UpdateSMContext` Request.** For each PDU Session to be handed over, the AMF sends an `Nsmf_PDUSession_UpdateSMContext` Request to the corresponding SMF, indicating a handover preparation and providing the Target ID. The SMF determines whether the existing UPF(s) can serve the target or whether a new intermediate UPF is needed [1].

3. **SMF ↔ UPF: N4 Session Modification (prepare).** The SMF sends an N4 Session Modification Request to the UPF to prepare the data-plane path (e.g., set up forwarding tunnels for indirect forwarding). The UPF acknowledges with an N4 Session Modification Response [1, 22].

4. **SMF → AMF: `Nsmf_PDUSession_UpdateSMContext` Response.** The SMF returns the N2 SM information for the target (CN Tunnel Info, QoS flow information, and any indirect forwarding tunnel information). If the SMF could not prepare the PDU Session, it indicates failure [1].

5. **AMF → T-RAN: Handover Request.** The AMF sends an NGAP *Handover Request* to the T-RAN carrying the UE security context, the UE Aggregate Maximum Bit Rate, the PDU Session resource list (with CN Tunnel Info and QoS profiles), and the Source-to-Target Transparent Container [1, 19].

6. **T-RAN → AMF: Handover Request Acknowledge.** The T-RAN performs admission control. If it can accommodate the UE, it allocates resources and replies with an NGAP *Handover Request Acknowledge* containing the Target-to-Source Transparent Container (with the RRC Handover Command), the list of successfully established PDU Sessions (with DL N3 AN Tunnel Info), and any failed PDU Sessions [1, 19].

7. **AMF → SMF: `Nsmf_PDUSession_UpdateSMContext` Request.** For each successfully set up PDU Session, the AMF forwards the T-RAN's N3 AN Tunnel Info to the SMF [1].

8. **SMF ↔ UPF: N4 Session Modification (path switch).** The SMF sends a second N4 Session Modification Request to the UPF with the T-RAN tunnel information, preparing the downlink path switch. The UPF acknowledges [1, 22].

9. **SMF → AMF: `Nsmf_PDUSession_UpdateSMContext` Response.** The SMF confirms that the data-plane preparation is complete. If indirect data forwarding is enabled, the response includes the forwarding tunnel information [1].

The AMF then sends an NGAP *Handover Command* to the S-RAN, carrying the Target-to-Source Transparent Container and the forwarding information. Upon receipt, the S-RAN initiates the execution phase [1, 19].

Source-to-Target Transparent Container The *Source-to-Target Transparent Container* carries access-specific state from the source to the target node. In NG-RAN handover, it typically encapsulates RRC configuration, UE capabilities, and other radio-layer context. The AMF forwards this container without interpreting its contents, treating it as an opaque payload. This design allows direct transfer of access-layer state between RAN nodes while preserving core-network transparency.

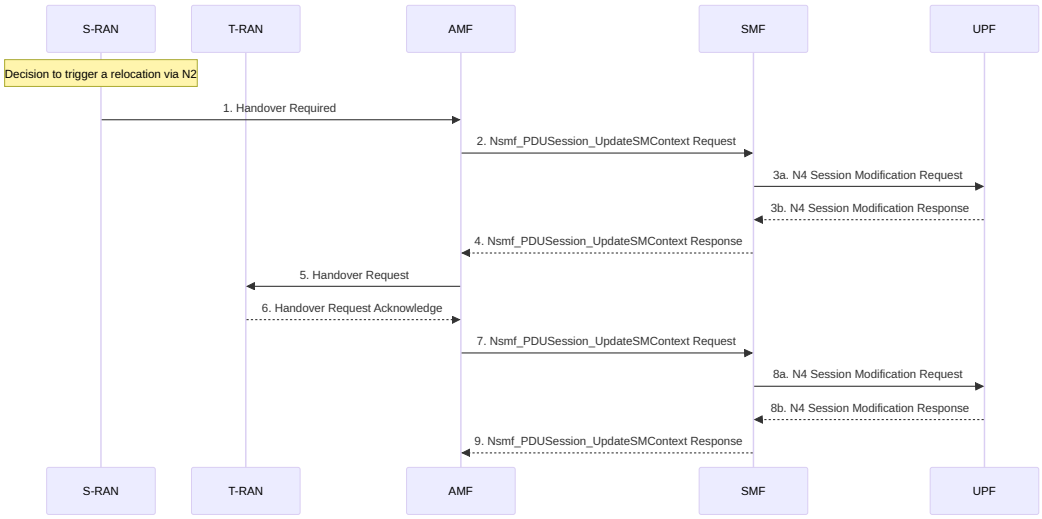

Figure 2.3: Preparation Phase of N2 Handover (3GPP TS 23.502, clause 4.9.1.3.2).

Execution Phase

The execution phase of the Inter NG-RAN node N2-based handover begins once the S-RAN receives the Handover Command from the AMF. The procedure is specified in 3GPP TS 23.502 (clause 4.9.1.3.3) [1] and illustrated in Figures 2.4 and 2.5. As for the preparation phase, the description below is a simplified version; the complete procedure, including AMF relocation, intermediate UPF insertion, secondary RAT reporting, and indirect forwarding cleanup, is detailed in 3GPP TS 23.502 [1].

1. **AMF → S-RAN: Handover Command.** The AMF forwards the NGAP *Handover Command* to the S-RAN, carrying the Target-to-Source Transparent Container and the forwarding information for each PDU Session.

2. **S-RAN → UE: Handover Command.** The S-RAN forwards the Target-to-Source Transparent Container to the UE, instructing it to perform the handover to the target cell.

3. **S-RAN → T-RAN: Uplink RAN Status Transfer.** The S-RAN sends an *Uplink RAN Status Transfer* directly to the T-RAN, preserving PDCP sequence continuity.

4. **AMF → T-RAN: Downlink RAN Status Transfer.** The AMF forwards the *Downlink RAN Status Transfer* to the T-RAN.

5. **Data Forwarding.** During the transition, the S-RAN forwards buffered downlink data towards the T-RAN via *direct forwarding* (step 5a) or *indirect forwarding* through the UPF (step 5b), depending on connectivity between source and target.

6. **UE → T-RAN: Handover Confirm.** Once the UE has synchronized to the target cell, it sends a *Handover Confirm* to the T-RAN. User-plane data can now flow between the UE and the T-RAN.

7. **T-RAN → AMF: Handover Notify.** The T-RAN notifies the AMF of the successful handover.

8. **AMF → SMF: `Nsmf_PDUSession_UpdateSMContext` Request.** For each PDU Session, the AMF sends a *Handover Complete indication* to the SMF.

9. **SMF ↔ UPF: N4 Session Modification.** The SMF instructs the UPF to switch the downlink data path to the T-RAN. The UPF sends *end marker* packets on the old path and begins forwarding to the target.

10. **SMF → AMF: `Nsmf_PDUSession_UpdateSMContext` Response.** The SMF confirms the path switch is complete.

11. **Mobility Registration Update.** The UE performs a Mobility Registration Update; since the AMF already holds the UE context from the handover, several registration steps are skipped.

12. **AMF ↔ S-RAN: UE Context Release.** The AMF releases the UE context on the S-RAN, freeing all associated resources.

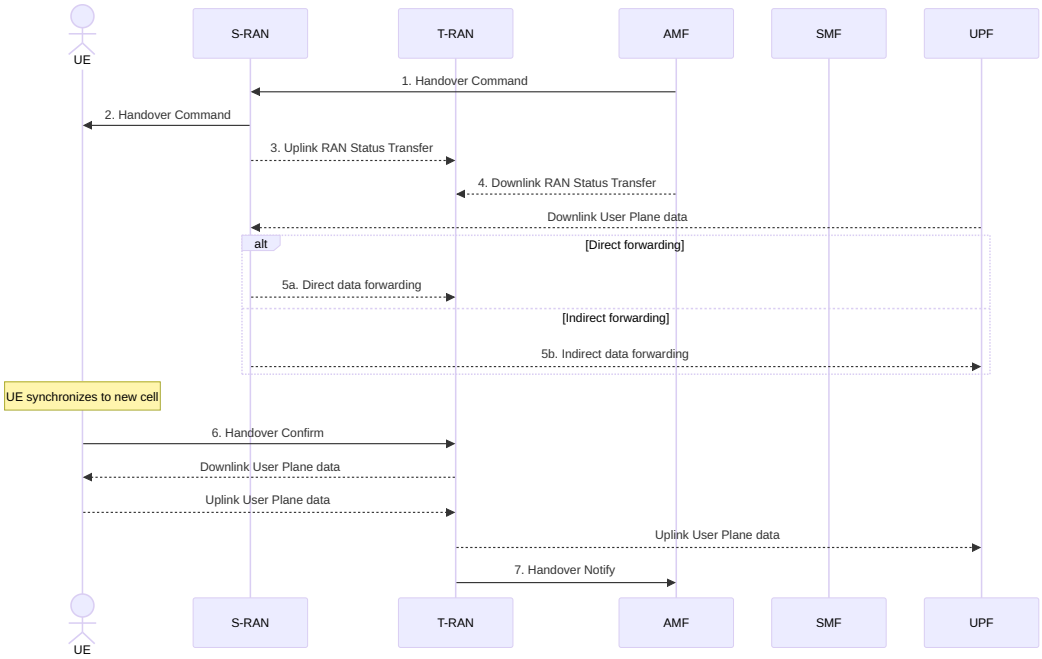

Figure 2.4: Execution Phase of N2 Handover – Part 1 (based on TS 23.502, clause 4.9.1.3.3 [1]).

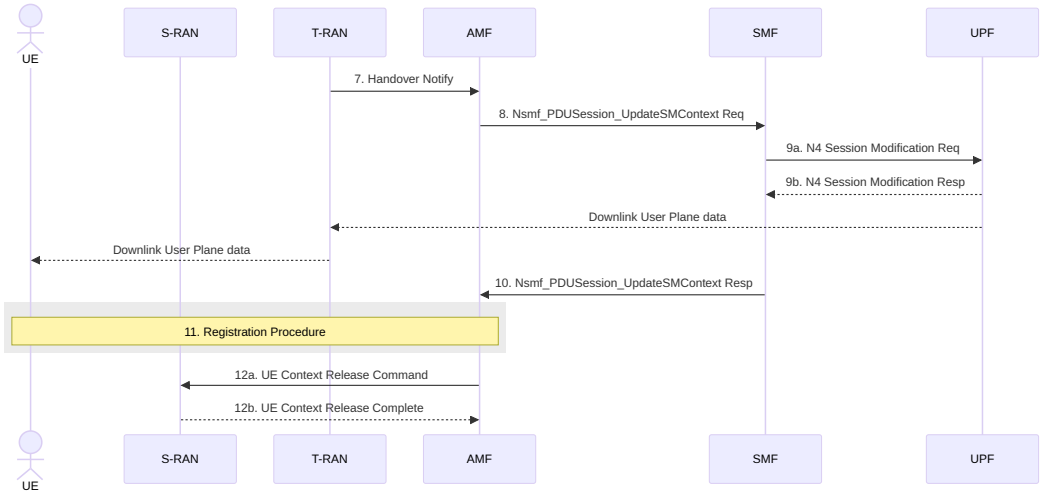

Figure 2.5: Execution Phase of N2 Handover – Part 2 (based on TS 23.502, clause 4.9.1.3.3 [1]).

2.3 Non-3GPP Access in 5G

Unlike previous generations, 5G treats non-3GPP access (e.g., Wi-Fi, fixed broadband) as a first-class option: a UE can reach the *same 5G Core* through a non-3GPP network while still benefiting from the full set of 5GC procedures (authentication, session management, QoS, and mobility) [2, 6]. This is the foundation of the work presented in this thesis, where the UE connects to the 5GC over Wi-Fi connection.

From a system perspective, non-3GPP access supports the following design motivations and use cases:

- **WLAN integration and selective offload**: 5GC connectivity can be provided through WLAN and other non-3GPP technologies, allowing traffic to be carried over non-3GPP links when beneficial (e.g., for indoor coverage or capacity relief) while preserving 5GC anchoring and session semantics [2, 1].
- **Secure connectivity over untrusted networks**: when the access is not trusted, the UE establishes secure tunnel(s) towards the 5GC via the N3IWF so that both control-plane and user-plane traffic are protected while traversing the untrusted access [2, 1, 6].
- **Operator/partner-managed trusted access**: trusted non-3GPP access (TNAN) enables 5G connectivity via a trusted access domain with AAA integration and a dedicated gateway (TNGF), supporting controlled onboarding and policy-driven use of trusted WLAN deployments [2, 1].
- **Support for devices not fully 5G-capable over WLAN**: 3GPP also defines architectural enhancements that allow devices that do not support 5GC NAS signalling over WLAN to access 5GC via *trusted WLAN* with assistance functions (e.g., TWIF) [2].

Trusted vs. Untrusted Non-3GPP Access 3GPP specifies two interworking models for integrating non-3GPP access into 5GS, depending on whether the access network is within the operator's trust [2]. This thesis adopts the **untrusted** model, using the **N3IWF** as the gateway between the Wi-Fi access and the 5GC, since the testbed simulates a publicly accessible Wi-Fi scenario where the access network is not operator-controlled.

- **Trusted non-3GPP access (TNAN)**: the access network is trusted by the operator and provides *trusted connectivity* to a PLMN. The

TNAN connects to the 5GC via a **Trusted Non-3GPP Gateway Function (TNGF)**. The UE establishes a **secure NWt connection** towards the TNGF; NAS messages between UE and AMF are transferred via this NWt connection. Trusted access commonly relies on an AAA relationship between the trusted access point (TNAP) and the TNGF and may leverage EAP-based procedures at L2, while 5GC registration remains protected via the secure connection [2, 1].

- **Untrusted non-3GPP access**: the access network is not trusted by the operator. The UE establishes **secure tunnel(s)** (NWu) towards the **Non-3GPP Interworking Function (N3IWF)** so that control-plane and user-plane traffic is transported securely over the untrusted access. During registration, NAS messages may be carried using **EAP-5G** encapsulation (for NAS transport, not as the authentication method itself), and subsequent NAS signalling is carried over the established secure association [2, 1, 6].

Table 2.3: Comparison of Trusted and Untrusted Non-3GPP Access

Aspect	**Trusted (TNAN)**	**Untrusted**
Gateway NF	TNGF (Trusted Non-3GPP Gateway Function)	N3IWF (Non-3GPP Interworking Function)
Security	Access network security (e.g., WPA3)	IPsec tunnel (IKEv2 + ESP)
NAS Transport	Via TNGF (NWt interface)	Via IPsec tunnel (NWu interface)
Typical Use Case	Operator-managed Wi-Fi, enterprise WLAN	Public Wi-Fi, home broadband, VPN scenarios
UE Requirements	Support for trusted access procedures	IKEv2/IPsec stack, EAP-5G support

2.3.1 Non-3GPP Interworking Function (N3IWF)

The **Non-3GPP Interworking Function (N3IWF)** enables a UE to access the *5G Core (5GC)* over an *untrusted* non-3GPP access network (e.g., public Wi-Fi or residential broadband), by acting as a secure gateway between the UE and the 5GC [2, 1, 23]. In particular, the N3IWF terminates the UE-facing security tunnel(s) on the **NWu** reference point and presents itself towards the 5GC as a **5G-AN node** supporting **NGAP** over **N2** and user-plane tunneling over **N3** [2].

Role in the 5G architecture From the core's perspective, the N3IWF is an access node: it provides N2 (control plane) toward the AMF and N3 (user plane) toward the UPF, while protecting UE-to-core traffic via NWu across the untrusted access [2, 23]. It is the first operator-controlled element on the path from the Wi-Fi network into the 5GC.

Key functionalities TS 23.501 lists the following N3IWF functions [2]:

- **NWu security termination**: termination of IKEv2/IPsec with the UE and establishment of IPsec SAs.
- **N2/N3 termination**: termination of N2 (control plane) and N3 (user plane) towards the 5GC.
- **NAS relaying**: relay of uplink/downlink NAS (N1) between UE and AMF.
- **PDU session and QoS signaling support**: handling of N2 SM information (from SMF via AMF) related to PDU sessions and QoS.
- **User-plane relaying**: decapsulation/encapsulation between IPsec (NWu) and GTP-U (N3) and forwarding between UE and UPF.
- **QoS/traffic marking**: enforcement of QoS/marking policies (e.g., DSCP) on tunnel traffic, including the possibility of outer-tunnel marking policies that differ from inner packets.
- **Mobility-related support**: local mobility anchoring for untrusted non-3GPP access and support for AMF selection.

The N3IWF's interfaces (NWu, N2, N3) and their protocol realizations are summarized in Table 2.1 and illustrated in Figure 2.6.

NAS Transport over NWu The UE registration over untrusted non-3GPP access uses two logical phases [6, 23, 1]:

1. **NAS during IKEv2 authentication (EAP-5G encapsulation)**: NAS messages are transported between UE and AMF by encapsulating NAS PDUs into EAP-5G messages carried during IKEv2 signaling. Importantly, EAP-5G acts as an encapsulation/transport method for NAS, while primary authentication remains 5G authentication (e.g., 5G-AKA or key-generating EAP methods) driven by the AMF/AUSF. EAP-5G is a vendor-specific EAP method (type 255, per RFC 3748 [25]) defined in TS 24.502 [6].

2. **NAS after tunnel establishment**: after the *signalling IPsec SA* (first Child Security Association) is established, subsequent NAS messages are carried inside the secure tunnel (over the signalling IPsec SA) using the NAS transport procedures defined for non-3GPP access.

User Plane and QoS Flow Mapping For each PDU Session established via N3IWF, the UE and N3IWF create one or more **user-plane IPsec Child SAs** using IKEv2 Create_Child_SA exchanges [1]. The 5GC provides the QoS information (including QFI assignments) to enable per-QoS-flow handling. In particular [1]:

- **Uplink**: the UE encapsulates uplink PDUs in **GRE**, carrying the associated **QFI** in the GRE header, and sends them via the IPsec Child SA bound to that QFI.
- **Downlink**: when the N3IWF receives a downlink PDU from the UPF on N3, it uses the **QFI** and PDU session identity to select the correct IPsec Child SA, encapsulates the PDU in GRE, and forwards it over NWu (optionally including an RQI for reflective QoS).

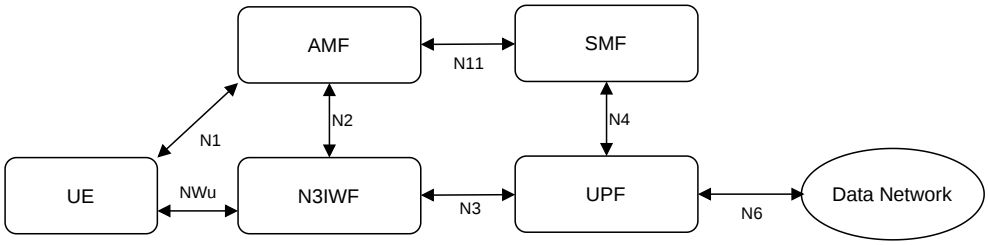

Figure 2.6: N3IWF positioning and interfaces for untrusted non-3GPP access (based on TS 23.501 [2]).

UE-side perspective (N3IWUE) In this thesis, the term **N3IWUE** denotes a standard 5G UE operating over untrusted non-3GPP access via the N3IWF. It is not a separate 3GPP device class; rather, it highlights the UE behaviors specific to non-3GPP access: IKEv2/IPsec and EAP-5G instead of RRC/PDCP [1, 6]. Registration is tightly coupled to IKEv2 establishment: the UE performs N3IWF selection, initiates IKE_SA_INIT and IKE_AUTH with EAP-5G encapsulating NAS messages, and upon successful authentication establishes the signalling IPsec SA. The N3IWF assigns an inner IP address for reliable NAS transport over TCP [6, 1] (Figure 2.7). For each PDU Session, the N3IWF creates one or more user-plane IPsec

Child SAs via CREATE_CHILD_SA; the UE maps QoS flows to Child SAs and encapsulates uplink PDUs in GRE with the QFI in the header [1, 6]. Figures 2.8 and 2.9 illustrate the control-plane and user-plane signalling for PDU Session establishment over untrusted non-3GPP access.

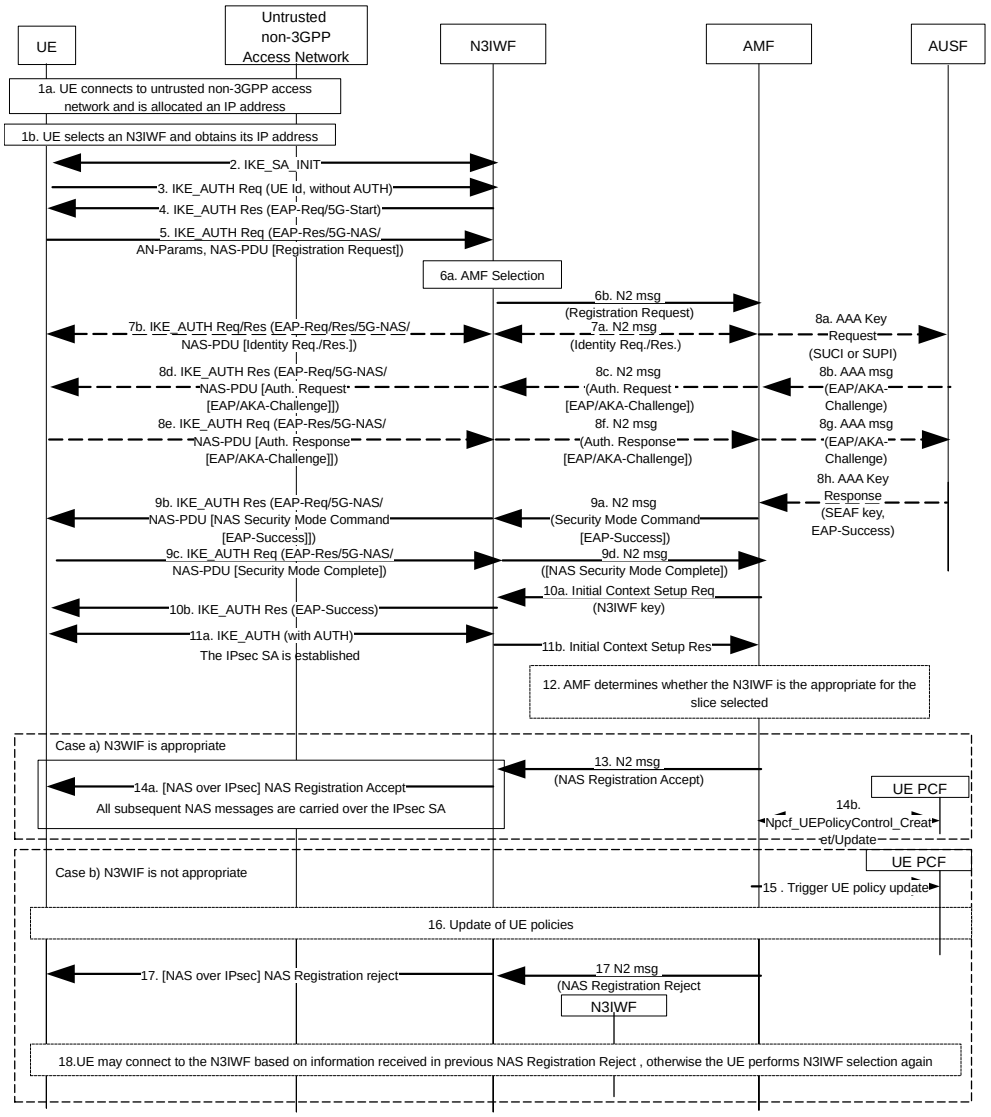

Figure 2.7: Registration procedure for untrusted non-3GPP access (TS 23.502 [1], Fig. 4.12.2-2).

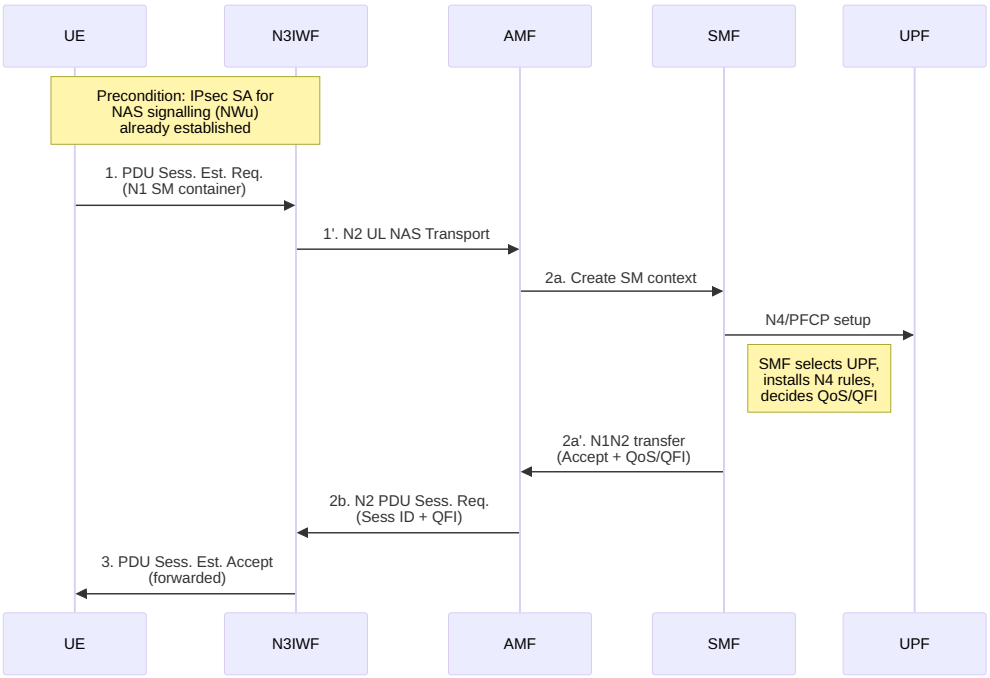

Figure 2.8: Control-plane signalling for PDU Session establishment over untrusted non-3GPP access (based on TS 23.502 [1]).

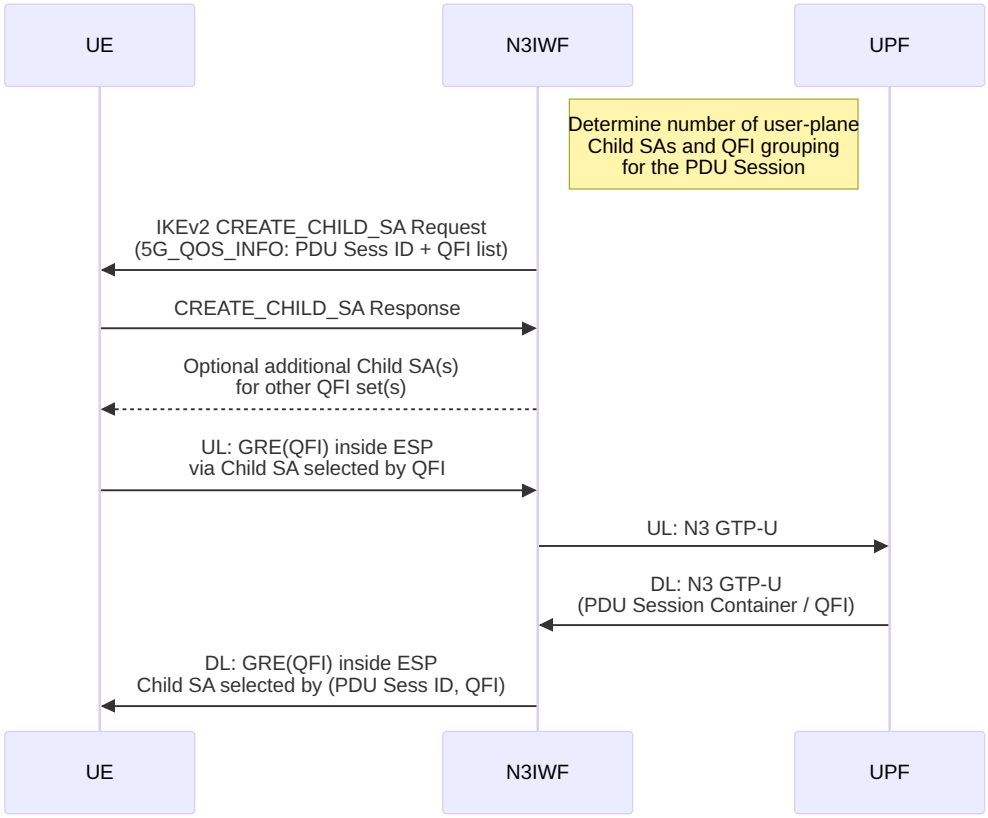

Figure 2.9: User-plane IPsec Child SA creation and QFI-to-Child-SA mapping for untrusted non-3GPP access (UE–N3IWF), with N3 GTP-U towards the UPF (based on TS 23.502 [1]).

2.3.2 IKEv2, IPsec, and MOBIKE

Since the UE-to-N3IWF path crosses an untrusted network, all traffic is protected by **IPsec** tunnels, negotiated and managed through **IKEv2** [26]. The **MOBIKE** extension [27] adds the ability to update tunnel endpoints when the UE changes IP address during handover.

Why IPsec is Required

In 3GPP access, security is provided at the air interface level: PDCP encryption and integrity protection secure the radio link, and the RAN is typically within the operator's trusted domain. In contrast, untrusted non-3GPP access traverses networks outside the operator's control.

IPsec addresses this by creating an **encrypted tunnel** between the UE and the N3IWF [28]:

- **Confidentiality**: ESP encryption (e.g., AES-GCM) prevents eavesdropping on the untrusted network
- **Integrity**: ESP authentication detects any tampering with packets
- **Authentication**: IKEv2 ensures mutual authentication between UE and N3IWF
- **Anti-replay**: Sequence numbers prevent replay attacks

IKEv2 Protocol Overview

IKEv2 (RFC 7296) [26] is the protocol used to establish and manage IPsec Security Associations (SAs). It establishes an IKE Security Association and negotiates one or more IPsec Child SAs that protect signaling and user-plane traffic between the UE and the N3IWF. In the non-3GPP access case, authentication is performed via EAP-5G, with NAS messages encapsulated within IKEv2 exchanges.

IPsec Child SAs

After IKEv2 authentication completes, one or more IPsec **Child Security Associations (Child SAs)** are established to protect actual traffic using ESP [29].

In the non-3GPP access scenario, two types of Child SAs are relevant:

- **Signaling SA**: protects NAS signaling exchanged between the UE and the N3IWF after registration.

- **User-plane SAs**: typically one per PDU session, carrying user traffic between the UE and the N3IWF.

Each Child SA defines the cryptographic parameters and traffic selectors that determine which flows are protected. In the N3IWF architecture, QoS flows are mapped to IPsec-protected tunnels rather than radio bearers, making the IPsec state a critical element for session continuity during mobility.

MOBIKE: IKEv2 Mobility and Multihoming

MOBIKE (IKEv2 Mobility and Multihoming Protocol, RFC 4555) [27] extends IKEv2 by allowing an established IKE Security Association (SA) to survive changes in the outer IP address of a peer. This capability is particularly relevant in untrusted non-3GPP access scenarios, where the UE may frequently change its point of attachment (e.g., between different Wi-Fi networks).

Without MOBIKE, an IP address change would require a complete IKEv2 re-negotiation, re-establishment of all associated IPsec Child SAs, and a new NAS-level mobility procedure to reactivate the PDU sessions. Such a process introduces significant latency and disrupts user-plane continuity.

MOBIKE addresses this limitation by enabling peers to update the transport addresses associated with an existing IKE SA through protected signaling exchanges, without recreating the underlying security associations. As a result, both the IKE SA and its Child SAs remain valid, allowing ongoing sessions to continue despite changes in the outer IP address.

However, MOBIKE assumes that both endpoints already share the relevant IPsec state. It does not define mechanisms for transferring Security Association state between different gateways. Consequently, while MOBIKE enables mobility across IP address changes, it does not by itself solve the problem of inter-gateway handover in non-3GPP access, where IPsec state continuity across distinct N3IWF instances must be explicitly addressed.

2.4 O-RAN Architecture

The **Open Radio Access Network (O-RAN)** initiative, led by the O-RAN Alliance [7], promotes a disaggregated, vendor-interoperable, and intelligent RAN architecture. Three guiding principles distinguish O-RAN from the traditional monolithic RAN [3]:

- **Openness**: standardized, open interfaces (E2, A1, O1, O2, Open Fronthaul) replace proprietary inter-component protocols, enabling multi-vendor deployments.

- **Disaggregation**: RAN functions are decomposed into modular components (O-CU-CP, O-CU-UP, O-DU, O-RU) that can be independently deployed and scaled.
- **Intelligence**: programmable controllers (Non-RT RIC, Near-RT RIC) inject data-driven optimization into the RAN control loop, using telemetry, machine learning, and policy-based reasoning.

These principles enable a new paradigm where third-party logic, packaged as xApps on the Near-RT RIC or rApps on the Non-RT RIC, can observe, decide, and act on RAN behavior in near-real time, without requiring modifications to the RAN nodes themselves. Figure 2.10 provides an overview of the O-RAN logical architecture and its main interfaces.

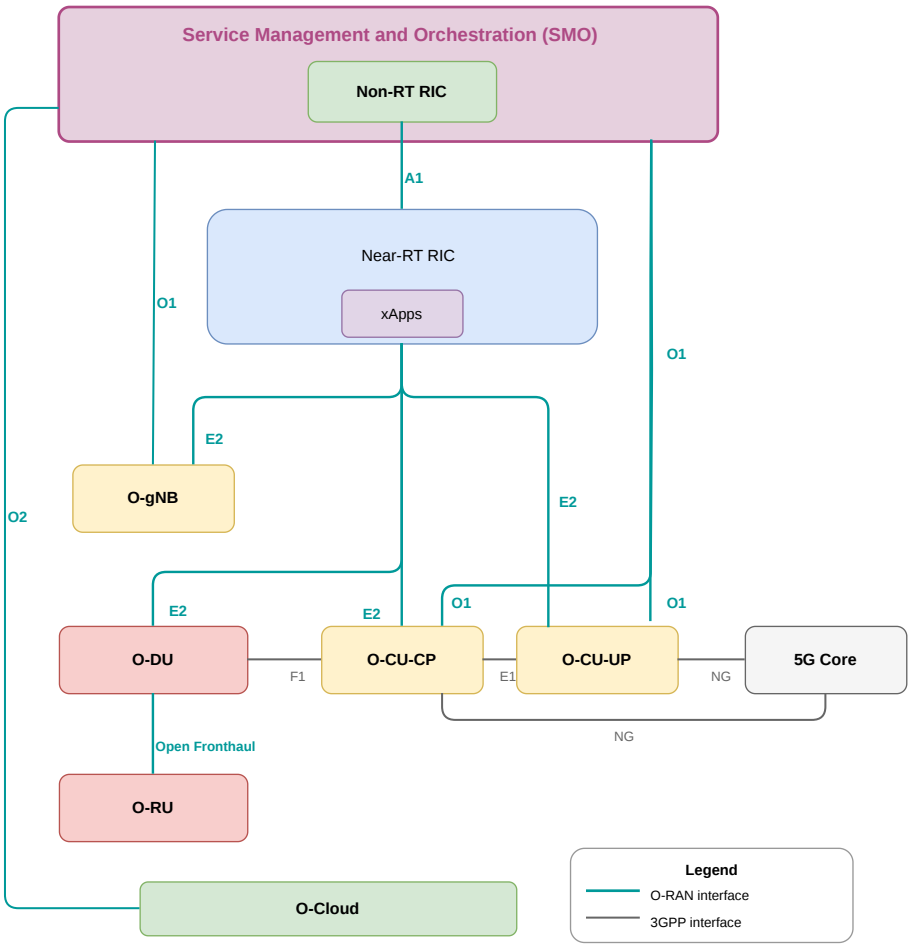

Figure 2.10: O-RAN logical architecture overview showing the main components and interfaces (adapted from O-RAN Architecture Description [3]). O-RAN-defined interfaces are shown in teal; 3GPP-defined interfaces in gray.

2.4.1 RAN Intelligent Controllers

O-RAN introduces two programmable controllers. The **Non-RT RIC** operates at timescales above one second: it manages A1 policies, trains ML models on historical data, and pushes enrichment information to the Near-RT RIC. The **Near-RT RIC**, operating between 10 ms and 1 s, is the component relevant to this thesis. It hosts **xApps**, containerized applications that receive telemetry from E2 Nodes and issue control actions back to them over the E2 interface. This observe–decide–act loop is the mechanism through

which the handover xApp developed in this work controls the N3IWF.

2.4.2 O-RAN Interfaces

O-RAN defines several key interfaces:

Table 2.4: O-RAN Key Interfaces

Interface	**Endpoints**	**Purpose**
E2	Near-RT RIC ↔ E2 Node	Near-real-time control and telemetry (E2AP, E2SM)
A1	Non-RT RIC ↔ Near-RT RIC	Policy, ML models
O1	SMO ↔ O-RAN NFs	FCAPS management
O2	SMO ↔ O-Cloud	Cloud infrastructure management
Open Fronthaul	O-DU ↔ O-RU	Lower layer split (eCPRI-based)

These interfaces are also illustrated in Figure 2.10, which shows their placement within the O-RAN logical architecture.

2.4.3 Near-RT RIC Architecture

Internally, the Near-RT RIC [3] consists of several cooperating components, illustrated in Figure 2.11. The **E2 Termination** manages the SCTP associations with E2 Nodes, terminates E2AP sessions, and routes messages between xApps and the RAN. The **Subscription Manager** tracks which xApps have subscribed to which E2 Node data, merging duplicate subscriptions to reduce E2 traffic. The **Messaging Infrastructure** (RMR, RIC Message Router) provides the internal message bus connecting these components to xApps. An **A1 Termination** exists for receiving Non-RT RIC policies, though it is not used in our scenario. Beyond the components shown in the figure, the platform also includes a Conflict Mitigation module (resolving concurrent control from multiple xApps), the SDL (Shared Data Layer, a Redis-backed key-value store for inter-xApp state), and the App Manager (lifecycle management of xApp containers).

2.4.4 E2 Node Concept

An **E2 Node** is any RAN element that connects to the Near-RT RIC via E2 [8]. Standard E2 Nodes include O-gNB, O-CU-CP, O-CU-UP, and O-

DU. Crucially, the concept is extensible: any element that implements E2AP and exposes relevant service models can act as an E2 Node. In this thesis, the N3IWF is exposed to the Near-RT RIC as an E2 Node by enabling E2AP support and relevant RAN Functions, as detailed in Chapter 4.2. Each E2 Node is identified by a Global E2 Node ID and advertises its supported RAN Functions during E2 Setup.

Positioning of the N3IWF in the O-RAN Context Although the N3IWF is defined as a 5G Core network function rather than a RAN element, it terminates both the N2 control-plane interface (NGAP signaling toward the AMF) and the N3 user-plane interface (GTP-U toward the UPF). From a functional perspective, it therefore handles both access-related control signaling and user-plane data forwarding, similarly to a RAN access node.

This architectural analogy suggests that, despite its classification as a core function, the N3IWF can be logically exposed as an access node from the perspective of near-real-time monitoring and control. This observation is relevant when considering its integration into the O-RAN closed-loop framework.

2.4.5 Closed-Loop Control

The combination of E2SM-KPM (telemetry) and E2SM-RC (control) enables a closed-loop cycle: E2 Nodes report metrics to the Near-RT RIC, xApps analyze them and decide on actions, and the resulting control requests are sent back to the E2 Nodes for execution, all within the 10 ms–1 s Near-RT RIC timeframe. In this thesis, the loop consists of KPM reports carrying per-UE throughput and UE count from each N3IWF, processed by the handover xApp, which triggers an RC Control Request to migrate a UE when it detects a load imbalance.

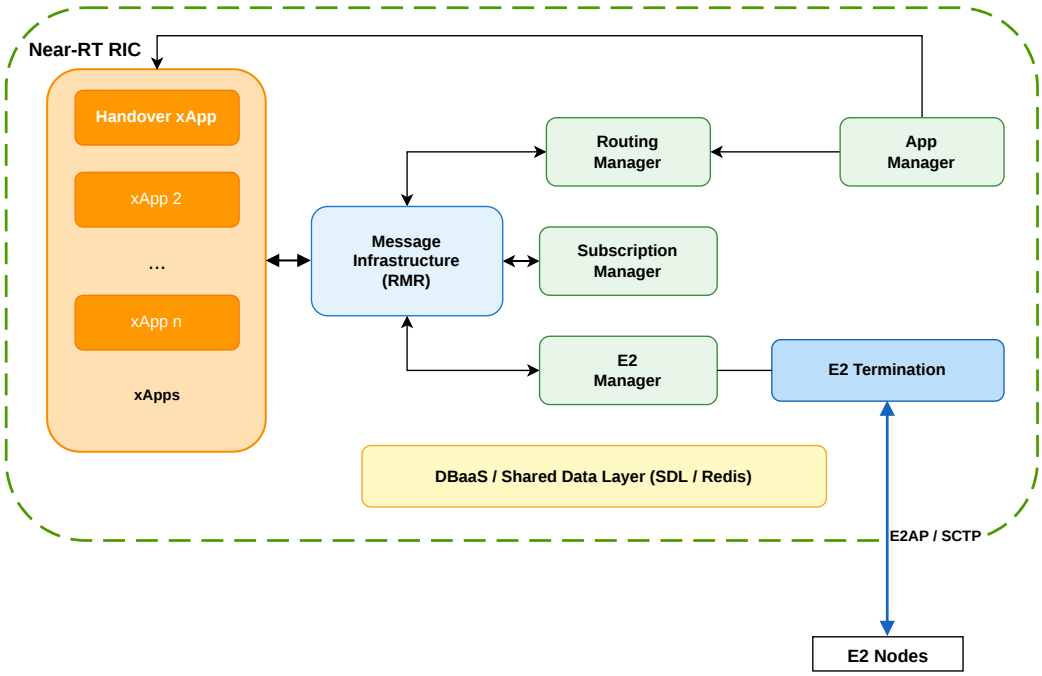

Figure 2.11: Near-RT RIC internal architecture showing key components and their interactions with E2 Nodes and xApps (based on O-RAN Architecture Description [3]).

2.5 E2 Interface

The **E2 interface** is the point-to-point channel between the Near-RT RIC and each E2 Node [8, 18]. It carries telemetry upward (performance metrics, event reports) and control commands downward (per-UE actions, policy installation), forming the foundation for closed-loop RAN optimization. Each E2 Node advertises its supported RAN Functions during E2 Setup, and the Near-RT RIC subscribes to the ones it needs.

Protocol Stack The E2 interface uses a layered stack: IP (network), SCTP (transport), E2AP (application-layer signaling), and E2 Service Models (E2SM) for payload semantics. In practice, E2AP messages (defined in ASN.1) are carried over SCTP with one association per node [18]. The stack is:

- **E2AP** (E2 Application Protocol) – defines signaling procedures (setup, subscription, control, etc.).
- **E2SM** (E2 Service Models)– payload formats for specific RAN functions (e.g. KPM, RC). Multiple E2SMs (e.g. E2SM-KPM for performance data, E2SM-RC for control) are multiplexed over E2AP [18].
- **SCTP** (Stream Control Transmission Protocol) - reliable transport (ordered streams) over IP.
- **IP**: Network layer transport.

This separation allows new service models to be defined without changing the underlying E2AP protocol. An E2 Node advertises which RAN Functions (E2SMs) it supports during E2 Setup, and the Near-RT RIC can then subscribe to those specific functions.

2.5.1 E2AP

The E2 Application Protocol (E2AP) provides the RIC–node signaling. E2AP is defined in ASN.1 and uses Aligned PER encoding (APER) for compact, extensible binary messages [8]. Every E2AP procedure has an Initiating Message (e.g. E2SetupRequest, SubscriptionRequest), followed by either a Successful Outcome (e.g. E2SetupResponse, SubscriptionResponse) or an Unsuccessful Outcome (failure message). Each message includes a procedure code, a criticality, and a value (the payload IEs). The ASN.1/APER rules ensure compatibility with 3GPP signaling standards (NGAP, X2AP, etc.) [4].

E2AP Procedures Table 2.5 lists the E2AP procedures used in this work.

Table 2.5: Key E2AP Procedures

Procedure	**Initiator**	**Type**	**Purpose**
E2 Setup	E2 Node	Request-Response	Establish E2 connection, advertise RAN Functions
RIC Subscription	Near-RT RIC	Request-Response	Subscribe to E2SM indications
RIC Indication	E2 Node	Unidirectional	Report metrics/events per subscription
RIC Control	Near-RT RIC	Request-Response	Execute control action on E2 Node
E2 Reset	Either	Request-Response	Reset E2 connection state

E2AP in Practice A typical E2 interaction flow:

1. **E2 Setup**: E2 Node connects and registers its RAN Functions (e.g., "I support E2SM-KPM v3 and E2SM-RC v1")
2. **RIC Subscription**: xApp (via Near-RT RIC) subscribes: "Send me KPM reports every 100ms for these metrics"
3. **RIC Indication (periodic)**: E2 Node sends measurement reports at the configured interval
4. **RIC Control**: xApp decides action needed and sends control request: "Trigger handover for UE X"
5. **RIC Control Ack**: E2 Node confirms action executed (or reports failure)

2.5.2 E2SM-KPM

E2SM-KPM [4] is the telemetry service model: it allows the Near-RT RIC to subscribe to periodic performance reports from E2 Nodes at node, cell, or UE granularity.

3GPP-aligned measurements and Wi-Fi mapping E2SM-KPM aligns its measurement names with 3GPP performance definitions [30]. In cellular RANs, many throughput- and volume-related counters are reported at the level of radio bearers (e.g., DRBs) or QoS flows, i.e., entities that do not exist in native Wi-Fi access. Therefore, exposing Wi-Fi telemetry through E2SM-KPM requires an explicit mapping from Wi-Fi-specific counters (e.g., per-station byte/packet statistics) to 3GPP-aligned measurement semantics, which is a key design aspect of this thesis.

Report Styles E2SM-KPM defines five **Report Styles**, each suited to different monitoring scenarios:

Table 2.6: E2SM-KPM Report Styles

Style	**Name**	**Description**
Style 1	E2 Node Measurement	Node-level or cell-level measurements aggregated across all UEs
Style 2	E2 Node Measurement for a Single UE	Measurements for a specific UE identified by UE ID
Style 3	Condition-based, UE-level	UE-level measurements reported when a condition is met (e.g., threshold crossing)
Style 4	Common Condition-based, UE-level	UE-level measurements for multiple UEs sharing a common condition
Style 5	E2 Node Measurement for Multiple UEs	Measurements for a set of UEs, aggregated or individual

Figure 2.12 illustrates the main subscription and indication flow; the full set of information elements for each style is detailed in the E2SM-KPM specification [4]. Each supported REPORT service style is composed of a well-defined set of E2AP and E2SM information elements, including the RIC Event Trigger Definition, the RIC Action Definition, the RIC Indication Header, and the RIC Indication Message.

E2SM-KPM supports exclusively Event Trigger Definition Format 1 (periodic, time-based) and a single Indication Header format (Format 1) conveying timestamp and optional metadata. Table 2.7 maps each Report Style to its corresponding E2SM information element formats.

Table 2.7: E2SM-KPM Styles and corresponding protocol structures

Report Style	**EventTrigger Definition**	**Action Definition**	**Indication Header**	**Indication Message**
Style 1	Format1	Format1	Format1	Format1
Style 2		Format2		
Style 3		Format3		Format2
Style 4		Format4		Format3
Style 5		Format5		

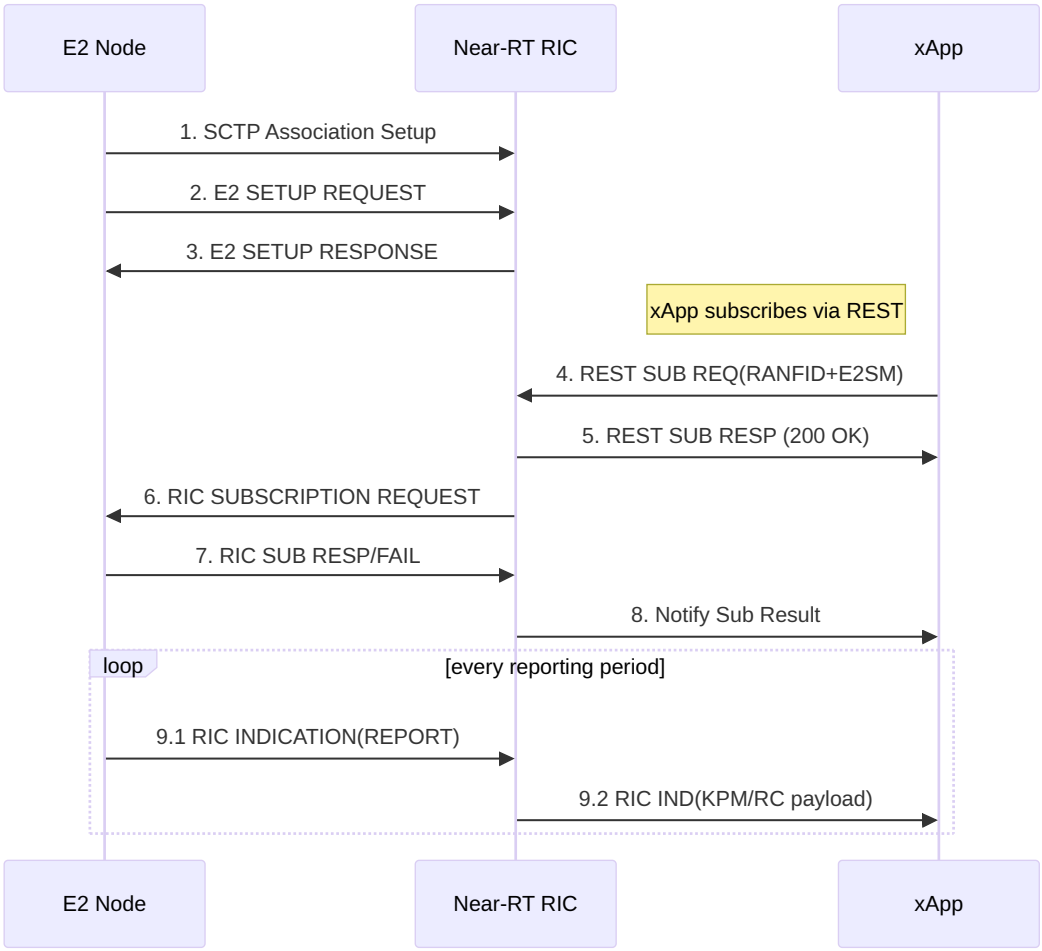

Figure 2.12: E2SM-KPM subscription and indication flow (based on O-RAN E2SM-KPM [4]).

2.5.3 E2SM-RC

E2SM-RC [5] is the control counterpart of KPM: it lets xApps trigger RAN procedures (e.g. handover, bearer modification), adjust parameters (scheduling weights, power settings), or install persistent policies. The E2 Node acknowledges each action with a success or failure outcome. RC supports two control types: *INSERT* for immediate one-shot commands, and *POLICY* for persistent rules that influence future E2 Node decisions.

Control Styles E2SM-RC defines several Control Styles, each corresponding to a RAN function category:

Table 2.8: E2SM-RC Control Styles

Style	**Name**	**Description**
Style 1	Radio Bearer Control	Control of DRB establishment, modification, release
Style 2	Radio Resource Allocation Control	Control of PRB allocation, scheduling parameters
Style 3	Connected Mode Mobility Control	Trigger handover, modify mobility parameters
Style 4	Radio Access Control	RRC connection control, access barring
Style 5	Dual Connectivity Control	Control of DC (Dual Connectivity) configurations
Style 6	Carrier Aggregation Control	Control of CA configurations

In this thesis, the relevant E2SM-RC functionality is **Control Style 3** (*Connected Mode Mobility Control*). The implemented handover xApp uses **Action ID 1** (*Handover Control*) to request the migration of a selected UE toward a target access node. In the proposed N3IWF-based scenario, the control message carries the target cell identity, which is interpreted by the E2-enabled N3IWF as the identifier of the destination N3IWF instance.

Each style has its own Action Definition formats and Control Message contents as defined in the O-RAN RC spec. For example, Style 1 allows the Near-RT RIC to issue commands like ”release DRB for UE X”.

Control Types: INSERT vs POLICY E2SM-RC distinguishes two control approaches:

INSERT Control. These are *immediate* one-off commands. The Near-RT RIC sends a RIC Control Request with a specific action to execute now.

The E2 node performs the action and returns a Control Outcome (success/-failure). Insert control is used for real-time interventions.

POLICY Control. These are *persistent policies*. The Near-RT RIC installs a high-level rule at the E2 node, influencing future decisions. For example, "prefer Cell B for handovers when Cell A is above X% load." The E2 node applies the policy internally; it does not execute a specific action immediately. Policy control is used for longer-term optimization.

Table 2.9: INSERT vs. POLICY Control Comparison

Aspect	**INSERT**	**POLICY**
Timing	Immediate execution	Applied to future decisions
Granularity	Specific action for specific entity	General rule for a class of decisions
Duration	One-time action	Persistent until modified/removed
Use Case	Real-time intervention	Strategic optimization
Example	"Handover UE X now"	"Route high-priority UEs to Cell B"

RIC Control Request and Outcome Control actions are sent via the E2AP RIC Control procedure. A *RIC Control Request* contains:

- **Control Header**: specifies the control style, target (e.g. UE ID, Cell ID), and action type.
- **Control Message**: contains style-specific parameters (e.g. target cell for handover, new scheduling weight).
- **Control Call Process ID**: an optional identifier to correlate related controls.

The E2 Node executes (or schedules) the action and responds with a *RIC Control Acknowledge* or *RIC Control Failure*, which includes:

- **Control Outcome**: indication of success or failure.
- **Control Outcome Details**: additional information (e.g. failure reason or applied values).

Figure 2.13 illustrates the main E2SM-RC control request flow; the complete set of information elements and additional subscription procedures are specified in the E2SM-RC standard [5].

Event Reporting In addition to active control, E2SM-RC supports event-based subscriptions. For example, an xApp can subscribe to RRC state changes, handover events, or radio link failures. When such an event occurs at the node (e.g. UE context release, handover completion), the node sends a RIC Indication. This allows the Near-RT RIC to monitor RAN events beyond periodic metrics, complementing E2SM-KPM's reports.

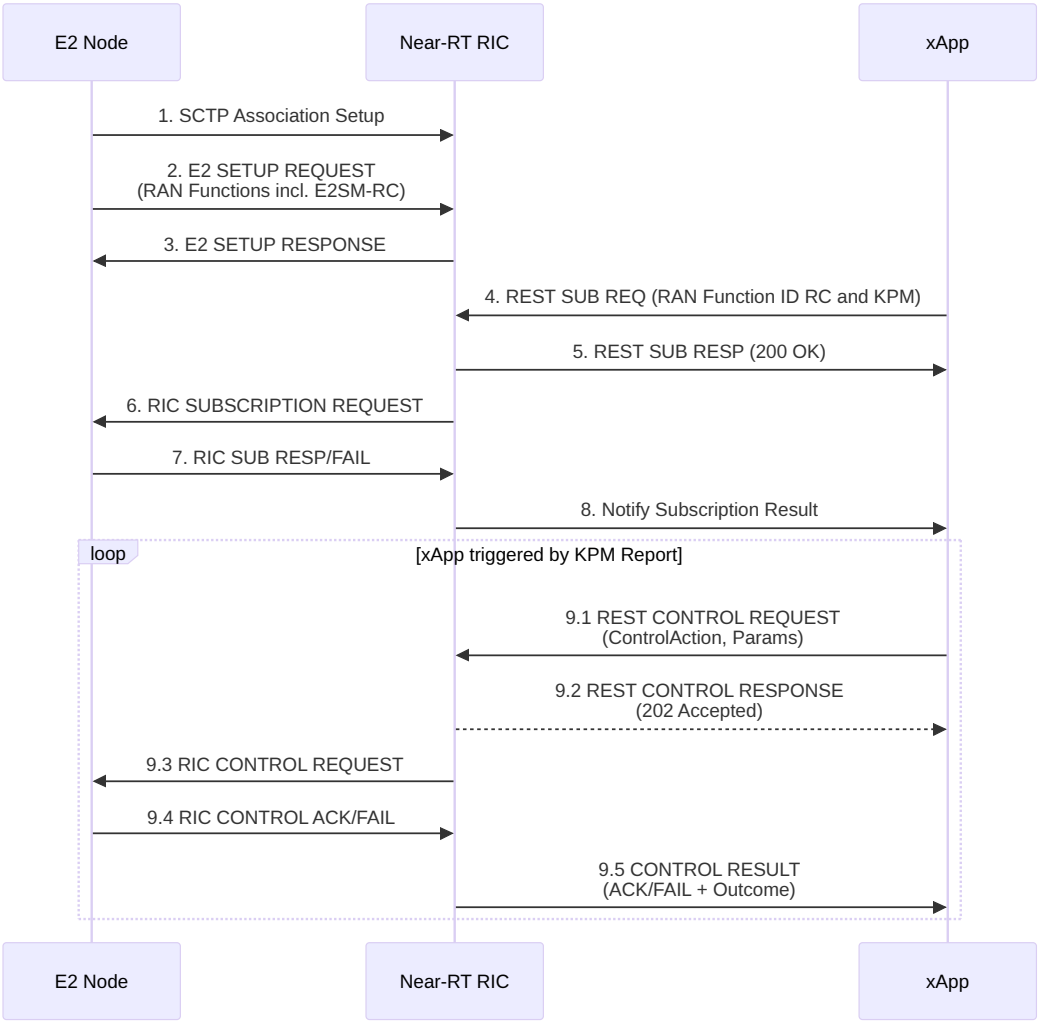

Figure 2.13: E2SM-RC subscription and control request flow (based on O-RAN E2SM-RC [5]).

2.6 xApps

This thesis focuses on xApps because the target use case, namely handover control for an E2-enabled N3IWF, belongs to the Near-RT RIC closed loop. As a result, xApps are the relevant programmable entities in this scenario, whereas rApps and Non-RT RIC functions are outside the scope of this work. An **xApp** is a containerized application running on the Near-RT RIC that implements a specific RAN optimization task [3]. Each xApp is single-purpose (e.g., load balancing, handover control, interference management), and multiple xApps can coexist on the same platform. At runtime, an xApp typically:

- Subscribes to RAN telemetry and events exposed by E2 Nodes via E2 Service Models (e.g., E2SM-KPM for performance metrics).
- Processes incoming data using rule-based logic, optimization algorithms, or machine learning models.
- Issues control commands to E2 Nodes using control-oriented service models such as E2SM-RC.
- Stores and retrieves state information through the Shared Data Layer (SDL).
- Optionally exchanges information with other xApps via SDL pub/sub mechanisms or REST interfaces.

This decouples the optimization logic from the RAN implementation: the same xApp can work with any E2 Node that exposes the required service models.

2.6.1 xApp Lifecycle and Deployment

An xApp is developed using an SDK provided by the OSC (O-RAN Software Community), packaged as a Docker image together with a descriptor (`config-file.json`), and deployed on the Kubernetes-based Near-RT RIC platform via a Helm chart generated by the `dms_cli` toolchain. At startup it initializes RMR (RIC Message Router) communication, creates E2 subscriptions, and begins processing indications. xApps communicate with E2 Nodes through the E2 Termination, manage subscriptions through the Subscription Manager, and can share state through the SDL key-value store. Typical use cases include traffic steering, handover optimization, interference management, and QoS enforcement.

2.6.2 xDevSM-Based xApps

Developing an xApp directly against E2AP and E2SM ASN.1 structures involves substantial boilerplate. The **xDevSM** (xApp Development Service Model) framework [31] wraps this complexity: the developer extends a framework class for a given service model (e.g., E2SM-KPM) and implements a callback invoked on each incoming E2 Indication. Subscription management, ASN.1 encoding/decoding, and RMR routing are handled internally. The resulting xApp is still packaged and deployed as a regular container on the Near-RT RIC.

The handover xApp developed in this thesis uses xDevSM for both the KPM subscription (telemetry) and the RC control path (handover trigger), which made it possible to iterate quickly on the control logic without dealing with low-level protocol details.

Chapter 3

Related Work

This chapter reviews the existing literature relevant to the contributions of this thesis and identifies the gaps that motivate the proposed solution. The discussion is organized into four thematic areas: the O-RAN architecture and E2 interface ecosystem, non-3GPP access integration in 5G, closed-loop RAN monitoring and control through the Near-RT RIC, and mobility management in non-3GPP access.

3.1 O-RAN Architecture and E2 Interface

The O-RAN Alliance has produced a rich ecosystem of specifications, and a growing body of academic work analyzes, extends, and evaluates it.

Polese et al. [32] provide the most comprehensive survey to date: they dissect the O-RAN architecture, its open interfaces (E2, A1, O1, O2), the Near-RT and Non-RT RIC platforms, and the security model. The survey catalogues published xApp demonstrations and identifies open research challenges, but every E2 Node discussed is a 3GPP RAN element (gNB, O-CU, O-DU). Bonati et al. [33] offer an earlier perspective on the convergence of softwarization, virtualization, and open interfaces in 5G, arguing that programmable RANs require standardized control loops; again, the scope is limited to 3GPP components.

On the operational side, D'Oro et al. [34] present *OrchestRAN*, a framework for multi-timescale orchestration of xApps and rApps across the Near-RT RIC hierarchy. Their experimental evaluation on the Colosseum testbed demonstrates closed-loop control through E2, but exclusively targets gNB-DU instances. Lacava et al. [35] extend this line with *ColO-RAN*, a traffic-steering xApp that uses reinforcement learning to optimize scheduling policies on O-DUs via E2SM-KPM telemetry and E2SM-RC control. Marinova

and Leon-Garcia [36] broaden the architectural discussion to "Beyond 5G" scenarios, surveying use cases from energy saving to network slicing, yet non-3GPP access does not appear among them.

Across this body of work, all E2 Node implementations and xApp demonstrations target 3GPP-compliant RAN elements. No prior work has extended the E2 interface to a non-3GPP interworking function or demonstrated E2SM-based monitoring and control of Wi-Fi access through the Near-RT RIC.

3.2 Non-3GPP Access Integration in 5G

The 3GPP specifications define the integration of non-3GPP access into the 5G system architecture (TS 23.501 [2]) and the detailed signaling procedures for untrusted access via the N3IWF (TS 24.502 [6]). These normative documents establish the reference points (N2, N3, NWu) and the registration flow (IKEv2/EAP-5G), but do not address observability or controllability of the N3IWF from an external orchestration layer.

Kunz and Salkintzis [37] cover the security aspects of non-3GPP access in 5G (EAP-5G authentication, IPsec tunnel establishment, the role of the N3IWF as a security gateway) and confirm the procedural complexity of the NWu interface. Lemes et al. [38] complement this with a hands-on tutorial implementing both trusted and untrusted access on a free5GC testbed. Neither work considers how the N3IWF could interact with RAN-level management frameworks such as O-RAN; in both cases the gateway is treated as a static element.

On the multi-path side, Wu et al. [39] survey transport-layer protocols (MPTCP, MPQUIC) for Access Traffic Steering, Switching and Splitting (ATSSS) in 5G, while 3GPP TR 23.793 [40] studies the architectural aspects of ATSSS support. These works address simultaneous use of 3GPP and non-3GPP paths at the transport layer, but not the coordination of non-3GPP access from the RAN intelligent controller.

Prados-Garzón et al. [41] discuss non-public network architectures where the N3IWF can serve as an on-ramp for enterprise Wi-Fi users, and Zampognaro et al. [42] demonstrate local edge services delivered through the N3IWF. Both works treat the N3IWF as a 5G Core function and do not explore its integration into the O-RAN framework. Mukute et al. [43] benchmark the control-plane performance of open-source 5G cores including free5GC, providing useful baseline data but no N3IWF-specific analysis.

While these works establish the architectural and procedural foundations for non-3GPP access in 5G, none of them addresses the integration of the

N3IWF into the O-RAN framework, nor do they provide mechanisms for standardized telemetry or RIC-driven control of the Wi-Fi access segment.

3.3 Closed-Loop RAN Control via the Near-RT RIC

This section reviews existing work on closed-loop RAN control through the Near-RT RIC, i.e. the observe (E2SM-KPM), decide (xApp), act (E2SM-RC) cycle, and shows that it has so far been confined to 3GPP RAN elements.

D'Oro et al. [34] and Lacava et al. [35] demonstrate the full observe–decide–act loop on the Colosseum testbed: *OrchestRAN* orchestrates xApps across time scales, while *ColO-RAN* applies reinforcement learning to traffic steering via E2SM-KPM and E2SM-RC. In both cases the E2 Nodes are O-DUs with native E2AP support.

Bashar et al. [44] present an experimental inter-DU load balancing system on an over-the-air 5G testbed. Their xApp uses E2SM-RC Control Style 3 (Connected Mode Mobility Control), Action ID 1 (Handover Control), the same service model action used in this thesis, to trigger handovers between O-DUs. This is one of the closest works to our approach in terms of E2SM-RC usage, but the target E2 Nodes are 3GPP O-DUs with native E2AP and standard RRC-based handover execution.

Vicario et al. [45] propose an intelligent mobility management framework for 5G O-RAN networks, where an xApp customizes handover parameters (e.g., A3 event thresholds) through the E2 interface to reduce ping-pong effects and unnecessary handovers. Barbosa and Dias [46] explore deep-learning-assisted mobility management for connected vehicles, where an xApp collects KPM telemetry and triggers handovers based on learned policies.

These works confirm that closed-loop control through the Near-RT RIC is viable, but they all assume native E2AP support on a 3GPP RAN element. The challenges specific to non-3GPP access (implementing E2AP as a sidecar agent, mapping Wi-Fi counters to 3GPP KPM metric names, and reconciling the Near-RT control-loop timing with the latency of N2 handover procedures) are not addressed. This thesis addresses these challenges.

3.4 Mobility and Handover in Non-3GPP Access

As a use case of the E2 integration, this thesis implements an N2-based handover between N3IWF instances. This section reviews the mobility landscape relevant to non-3GPP access.

Haghrah et al. [47] survey the handover management landscape in 5G-NR, covering intra- and inter-gNB handover, conditional handover, and dual-active-protocol-stack (DAPS) handover. Ullah et al. [48] extend the scope to heterogeneous networks (HetNets), including Wi-Fi offloading and vertical handover between cellular and Wi-Fi, but their treatment of the non-3GPP side is limited to high-level ATSSS-based steering rather than N2-level handover between N3IWF instances.

The MOBIKE protocol (RFC 4555 [27]) enables an IPsec peer to change its outer IP address without re-establishing the IKE Security Association. While MOBIKE is widely deployed for mobile VPN scenarios, its application to 5G non-3GPP handover has not been explored in the literature. The existing MOBIKE work focuses on address updates within a single security gateway; the scenario where a UE performs MOBIKE towards a *different* gateway that has pre-imported the IKE/IPsec state is, to our knowledge, novel.

Manolopoulos and Alevizaki [49] present a demonstration of multi-access 6G networks with user mobility considerations, where the N3IWF plays a role in enabling non-3GPP connectivity. Their work acknowledges the need for handover support between non-3GPP access points but does not implement an N2-based inter-N3IWF handover or address the challenge of IPsec state transfer.

The transfer of IPsec security association state between network elements during handover remains largely unexplored. While 3GPP RAN procedures define state transfer mechanisms for radio protocols, no equivalent mechanism has been systematically studied for IPsec-based non-3GPP access in the context of 5G mobility.

The N2-based handover between N3IWF instances, combining IPsec state synchronization with MOBIKE-based tunnel re-binding and orchestrated by the Near-RT RIC through the E2 interface, has not been addressed in prior work.

3.5 Summary and Positioning

The existing literature shows a clear divide: O-RAN research focuses exclusively on 3GPP RAN elements, while non-3GPP integration work remains within the scope of 5G Core procedures.

The O-RAN ecosystem is mature for 3GPP RAN elements: the E2 protocol, the service models (KPM, RC), and the xApp programming model are well understood and experimentally validated. Non-3GPP access integration in 5G is well specified by 3GPP, and open-source implementations such as free5GC provide working N3IWF stacks; however, no work exposes the N3IWF to the Near-RT RIC as a monitorable and controllable entity. The closed-loop control paradigm that O-RAN provides for gNBs (standardized telemetry collection, centralized decision-making in xApps, and RIC-issued control actions) has never been extended to non-3GPP access nodes. As a consequence, mobility in non-3GPP access is limited to session re-establishment or ATSSS-based path switching; no prior work integrates non-3GPP mobility procedures into the O-RAN closed-loop paradigm.

To the best of our knowledge, this is the first work to (a) integrate a non-3GPP interworking function as an O-RAN E2 Node with E2SM-KPM and E2SM-RC support, enabling standardized monitoring and control of Wi-Fi access through the Near-RT RIC, and (b) validate this integration through a concrete use case: an N2-based handover between N3IWF instances with IPsec state synchronization and MOBIKE-based execution, orchestrated in a closed loop by a Handover xApp.

Chapter 4

Enabling E2-Based Monitoring and Control of Non-3GPP Nodes: System Design

This chapter presents the architectural and design decisions underlying the system proposed in this thesis. The central design goal is to make a non-3GPP interworking function (N3IWF) visible and controllable within the O-RAN framework by integrating it as an E2 Node. This integration enables a complete E2 closed loop for RIC-driven N2-based handover between N3IWF instances, where E2SM-KPM provides the observation arm (Wi-Fi telemetry) and E2SM-RC provides the actuation arm (handover control).

4.1 Overall System Architecture

Figure 4.1 shows the overall system architecture. At the bottom, one or more non-3GPP UEs connect via Wi-Fi to Access Points associated with the N3IWF instances. Each UE establishes IKEv2/IPsec tunnels (NWu) through the Access Point to its serving N3IWF, which terminates the N2 interface towards the AMF and the N3 interface towards the UPF in the 5G Core. A single AMF serves both N3IWF instances, enabling intra-AMF N2-based handover. Alongside each N3IWF, a co-located E2 agent connects to the Near-RT RIC over E2AP/SCTP, exposing E2SM-KPM and E2SM-RC RAN Functions. Inside the Near-RT RIC, the E2 Termination and Subscription Manager handle the E2 protocol, while a Handover xApp implements the control logic.

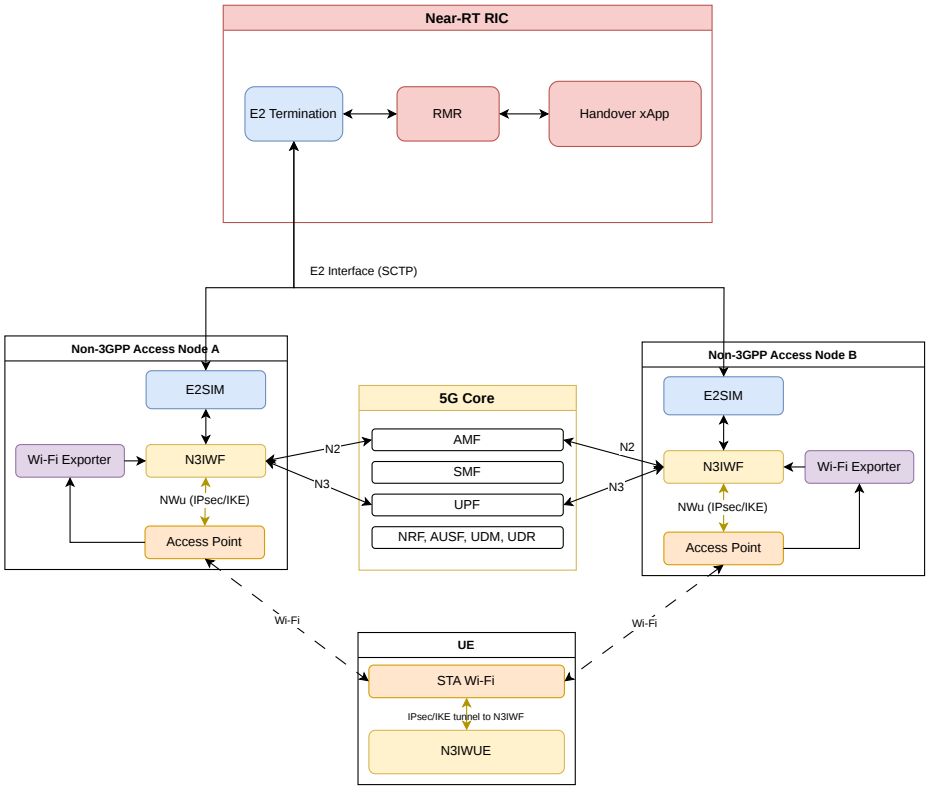

Figure 4.1: High-level system architecture. Each N3IWF instance is paired with a co-located E2 agent that exposes it to the Near-RT RIC as an E2 Node. The N3IWF connects to the 5G Core via N2/N3 and to the UE via IPsec tunnels (NWu) over Wi-Fi.

The data flow follows two paths. On the *monitoring path*, per-UE Wi-Fi metrics are collected at the Access Point, correlated with NGAP UE contexts at the N3IWF, and delivered to the xApp as E2SM-KPM Indications via the E2 agent. On the *control path*, the xApp issues an E2SM-RC Control Request to the E2 agent, which triggers the N3IWF to initiate the NGAP N2 handover procedure towards the AMF. Together, these two paths form the observe–decide–act closed loop: the xApp continuously receives KPM telemetry, evaluates load imbalance across the N3IWF instances, and when needed triggers a handover via RC.

4.2 Integrating N3IWF as an E2 Node

In the standard O-RAN architecture [8], each RAN component, whether a monolithic gNB or a disaggregated element (O-CU-CP, O-CU-UP, O-DU), acts as an E2 Node that connects to the Near-RT RIC over an SCTP association. The E2 Node implements E2AP and exposes one or more RAN Functions (e.g., E2SM-KPM, E2SM-RC); each E2 Node is identified by a `GlobalE2node-ID`.

The N3IWF does not natively implement E2AP. Rather than embedding E2AP support directly into the N3IWF codebase, a co-located E2 agent is deployed alongside each N3IWF instance in a sidecar pattern. The agent acts as the E2 Node on behalf of the N3IWF: it implements E2AP, connects to the Near-RT RIC, and exposes the E2SM-KPM and E2SM-RC RAN Functions. This sidecar approach keeps the N3IWF codebase free of O-RAN dependencies while still enabling full Near-RT RIC integration. The N3IWF's identity is presented to the Near-RT RIC as a `GlobalE2node-gNB-ID` derived from the N3IWF ID [8] as described in Section 4.2.2.

4.2.1 E2AP Communication Flow

For the N3IWF to be E2-compliant, the co-located E2 agent must support the E2AP procedures that govern the lifecycle of the E2 connection and the exchange of data and control messages with the Near-RT RIC [8]. The following E2AP procedures are required:

- **E2 Setup**: The E2 agent establishes an SCTP association with the E2 Termination in the Near-RT RIC and sends an *E2 Setup Request* carrying the `GlobalE2node-ID` (Section 4.2.2) and the list of supported RAN Functions (E2SM-KPM and E2SM-RC, with their advertised styles and capabilities). The Near-RT RIC accepts or rejects each RAN Function individually in the *E2 Setup Response*.

- **RIC Subscription (KPM path)**: An xApp subscribes to periodic KPM reports by sending a *RIC Subscription Request* specifying the Event Trigger (reporting period) and the Action Definition (report style, metric list, matching conditions). The E2 agent acknowledges and begins producing *RIC Indication* messages at the requested interval, each containing an E2SM-KPM Indication with per-UE measurement records.

- **RIC Control (RC path)**: An xApp triggers a control action by sending a *RIC Control Request* carrying an E2SM-RC Control Header

and Control Message (e.g., target cell identity for a handover). The E2 agent forwards the command to the N3IWF and returns a *RIC Control Acknowledge* (or *RIC Control Failure*) to the xApp. The actual procedure (e.g., N2 handover) executes asynchronously; the xApp monitors the outcome through subsequent KPM reports.

Internally, the E2 agent and the N3IWF exchange two categories of information: the N3IWF provides the agent with per-UE performance metrics and node identity/PLMN configuration (consumed by the KPM path), and the agent delivers control commands from the Near-RT RIC to the N3IWF (consumed by the RC path).

4.2.2 E2 Node Identity and RAN Function Registration

A key design decision is how the N3IWF identifies itself to the Near-RT RIC. The E2AP specification [8] defines `GlobalE2node-ID` as the following ASN.1 CHOICE:

```
GlobalE2node-ID ::= CHOICE {
    gNB        GlobalE2node-gNB-ID,
    en-gNB     GlobalE2node-en-gNB-ID,
    ng-eNB     GlobalE2node-ng-eNB-ID,
    eNB        GlobalE2node-eNB-ID,
    ...
}
```

Listing 4.1: E2AP GlobalE2node-ID CHOICE (E2AP v3.01 [8]).

None of these variants corresponds to a non-3GPP interworking function. Among the available choices, `gNB` is the only 5G-native variant: `en-gNB`, `ng-eNB`, and `eNB` all refer to LTE/EPC node types and carry EPC-specific identifiers (e.g., ECGI, TAI) that have no counterpart in the N3IWF configuration. The `GlobalE2node-gNB-ID` structure carries a `global-gNB-ID` field consisting of a PLMN identity and a 22–36-bit `gNB-ID` BIT STRING, which directly accommodates the N3IWF's numeric identifier within the same operator PLMN. It also defines two optional fields (`gNB-CU-UP-ID` and `gNB-DU-ID`) that are populated with placeholder values to satisfy the Near-RT RIC's E2 Manager registration logic [8] without carrying semantic meaning for a non-3GPP node. The E2 agent therefore presents the N3IWF using the `GlobalE2node-gNB-ID` variant. Table 4.1 shows how the N3IWF's configuration fields are mapped to the corresponding E2AP Information Elements.

From the Near-RT RIC's perspective, each N3IWF instance appears as a

Table 4.1: Mapping of N3IWF identity fields to the `GlobalE2node-gNB-ID` IE (E2AP v3.01).

N3IWF configuration	**E2AP IE**	**Notes**
MCC, MNC	`plmn-Identity` (3 bytes, BCD)	Standard 3GPP BCD encoding
N3IWF ID	`gNB-ID` (BIT STRING, 22 bits)	`gnb-ID` variant of `GNB-ID-Choice`
(not applicable)	`gNB-CU-UP-ID`	Placeholder (required by E2 Manager)
(not applicable)	`gNB-DU-ID`	Placeholder (required by E2 Manager)

standard gNB E2 Node. This reuse is intentional: because the Near-RT RIC dispatches subscriptions and control actions based on RAN Function IDs rather than on the node type, any existing xApp that consumes E2SM-KPM or issues E2SM-RC commands will interoperate with the N3IWF without modification.

Two RAN Functions are registered during the E2 Setup procedure:

- **E2SM-KPM** [4]: advertises Event Trigger Style 1 (periodic reporting), Report Style 4 (common condition-based, UE-level), and the list of supported metric names derived from the Wi-Fi-to-KPM mapping (Table 4.2).
- **E2SM-RC** [5]: advertises Event Trigger Style 4 (UE Information Change), Control Style 3 (Connected Mode Mobility Control) with Action ID 1 (Handover Control), and four control parameters encoding the target cell identity.

The Near-RT RIC accepts or rejects each RAN Function individually in the E2 Setup Response. Once accepted, the xApp can subscribe to KPM reports or issue RC control requests using the standard E2AP subscription and control procedures, with no awareness that the underlying node is a non-3GPP gateway.

4.3 Design Challenges

This section collects the main challenges and design choices that arose when integrating the N3IWF into the O-RAN framework and when adapting the

N2 handover procedure to non-3GPP access. Presenting them together provides the rationale behind the detailed mechanisms described in the subsequent sections.

4.3.1 Challenges in Enabling E2 on a Non-3GPP Node

Integrating the N3IWF as an E2 Node raised several challenges that do not arise with standard 3GPP E2 Nodes:

- **Mapping Wi-Fi metrics to 3GPP KPM names**: E2SM-KPM metric names are defined in terms of 3GPP measurement definitions (TS 28.552 [30]), which assume DRB-level radio bearer semantics. Wi-Fi station counters had to be mapped to the closest KPM equivalents to allow xApps to treat non-3GPP and 3GPP nodes uniformly.
- **Wi-Fi metric collection limitations**: Unlike 3GPP RAN elements, which expose structured performance counters through standardized interfaces, Wi-Fi access points provide only cumulative byte and packet counters per associated station. There is no event-driven notification mechanism; counters must be polled at a fixed interval. No equivalent of per-PRB utilization, CQI, or MCS statistics exists. Rate-based KPM metrics must therefore be derived by computing deltas between consecutive polls, introducing a dependency on the polling interval for metric granularity.
- **Sidecar E2 Node pattern**: Rather than embedding E2AP directly into the N3IWF, a co-located E2 agent acts as the E2 Node on its behalf. This keeps the N3IWF codebase free of O-RAN dependencies and allows the two components to evolve independently. The sidecar pattern also introduces an additional hop in the control path: upon receiving a RIC Control Request, the E2 agent forwards the command to the N3IWF; the N3IWF validates the request and responds immediately, after which the agent returns a RIC Control Acknowledge to the Near-RT RIC. The handover itself proceeds asynchronously and the xApp relies on subsequent KPM reports to observe the outcome. If the N3IWF does not return a timely response within the Near-RT RIC control window, the request is treated as failed and a *RIC Control Failure* is returned.

4.3.2 Challenges in Enabling N2 Handover over Non-3GPP Access

The standard Inter NG-RAN node N2-based handover procedure [1] was designed for mobility between 3GPP RAN nodes (gNBs). Adapting it to non-3GPP access via N3IWF required addressing three key design questions.

Handover mechanism selection Three alternatives were considered for enabling mobility between N3IWF instances:

1. **Xn-based handover**: In 3GPP RAN, Xn-based handover [24] allows direct context transfer between neighboring gNBs without involving the AMF in every step. However, the Xn interface is defined exclusively between gNBs (or disaggregated O-RAN components); no standardized Xn-equivalent exists between N3IWFs. Introducing a proprietary N3IWF-to-N3IWF interface would depart from the 3GPP architecture and would not benefit from any existing core-network support. This option was therefore discarded.

2. **Session re-establishment**: The UE could tear down the IPsec tunnel with the source N3IWF, associate with the target AP, and initiate a full IKEv2 negotiation with the target N3IWF from scratch. During IKE_AUTH, the UE performs a NAS Mobility Registration Update [1]; the AMF recognizes the UE via its 5G-GUTI and re-activates the PDU Sessions through the target N3IWF. While the UE is not treated as a new subscriber, the entire user-plane path is interrupted from the moment the source tunnel is released until the new IPsec tunnel, NAS registration, and PDU Session re-activation complete on the target.

3. **N2-based handover adaptation**: By reusing the 3GPP N2 handover procedure (with the N3IWF acting as a RAN node on N2), the handover can be coordinated through the AMF. The SMF and UPF are notified to switch the N3 tunnel endpoints, and the UE context is transferred between N3IWFs via the Source-to-Target Transparent Container, preserving ongoing PDU Sessions.

The N2 handover adaptation was chosen because it provides the best balance between standards compliance, service continuity, and feasibility. The N2 handover signaling (NGAP state machine in the AMF, SMF path-switch logic, UPF N4 session modifications) is already defined by 3GPP for inter-gNB mobility and can be reused with minimal changes. The design effort is therefore concentrated on the N3IWF-specific adaptations: IPsec state

transfer, MOBIKE-based execution, and downlink buffering. The detailed mechanism is described in Section 4.5.

Path switch timing In the standard 3GPP N2 handover procedure (TS 23.502 [1], clause 4.9.1.3), the UPF switches the downlink path to the target RAN *after* the Handover Notify, i.e. during the execution phase, once the target has confirmed that the UE has successfully accessed it. In such a *standard path switch*, the UPF continues sending downlink packets to the source RAN until the handover execution is complete. An *early path switch*, by contrast, redirects UPF downlink traffic to the target RAN's N3 tunnel endpoint during the preparation phase, before the UE has begun the handover execution.

The early path switch was chosen because, in the N3IWF scenario, once the UE begins the MOBIKE execution it updates its outer IP address to reflect the new Wi-Fi attachment point [27]. At that moment, the source N3IWF's IPsec outbound rules, which still reference the UE's old peer outer IP, can no longer deliver packets to the UE. The source N3IWF's IPsec tunnels remain technically alive (they are released only later, upon UE Context Release Command from the AMF after Handover Notify), but they are *functionally broken* for downlink delivery because the UE's outer IP has changed. With a standard path switch, downlink packets would continue arriving at the source during this window and be undeliverable. In a standard inter-gNB handover this problem is mitigated by direct forwarding over the Xn interface or by indirect forwarding tunnels through the UPF. Neither option is viable here: no Xn-equivalent interface exists between N3IWF instances, and routing indirect forwarding through the single UPF already serving the PDU Session would add load to the only user-plane node in the path without providing any buffering advantage over the target-side solution. The early path switch avoids these issues by redirecting UPF traffic to the target N3IWF before the execution phase begins, ensuring that downlink packets reach the node that will actually serve the UE once MOBIKE completes. The buffering strategy that accompanies this choice is detailed in Section 4.5.2.

Downlink buffering placement The 3GPP architecture allows the SMF to instruct the UPF to buffer downlink packets via PFCP Buffering Action Rules (BAR) [22]. However, the PFCP `SuggestedBufferingPacketsCount` IE is encoded as a `uint8`, capping UPF-side buffering at 255 packets. In our scenario, the handover takes on the order of hundreds of milliseconds (Wi-Fi switch + MOBIKE), during which a continuous downlink stream easily exceeds this limit. An N3IWF-side buffer avoids this capacity ceiling

without any core-network modifications. The buffer is placed on the *target* N3IWF because the early path switch redirects UPF traffic there during the preparation phase; the target is therefore the only node receiving downlink packets before the UE has completed the MOBIKE exchange (Section 4.5.2).

4.4 Monitoring: E2SM-KPM for Wi-Fi Access

With the N3IWF registered as an E2 Node, E2SM-KPM serves as the observation arm of the closed loop: it delivers per-UE Wi-Fi telemetry to the Near-RT RIC, enabling the handover xApp to detect load imbalances and decide when to trigger mobility. Because the E2 agent exposes the same E2SM-KPM RAN Function interface as a 3GPP E2 Node, this telemetry is also available to any other KPM-based xApp, whether designed for anomaly detection, capacity planning, or load monitoring, without modification and without needing to know that the underlying access is Wi-Fi rather than NR.

4.4.1 Choice of E2SM-KPM Report Style 4

E2SM-KPM v3 [4] defines five Report Styles (see Section 2.5.2). Report Style 4 ("Common Condition-based, UE-level") was selected because it provides per-UE measurement reports, essential for identifying which specific UEs are generating load on a given N3IWF, and supports condition-based filtering that limits reporting to UEs matching certain criteria (e.g., a specific S-NSSAI), reducing unnecessary traffic towards the Near-RT RIC.

4.4.2 Mapping Wi-Fi Metrics to KPM DRB-Level Metrics

Since the N3IWF operates over Wi-Fi (non-3GPP access), there are no native 3GPP DRB (Data Radio Bearer) metrics. Instead, the following mapping is used to translate Wi-Fi station statistics into 3GPP-aligned KPM metric names:

Each mapping approximates the TS 28.552 [30] definition under the constraints of non-3GPP access. For `DRB.UEThpDl` and `DRB.UEThpUl`, the specification defines throughput as the transmitted volume divided by the *active transmission time* (`ThpTimeDl`/`ThpTimeUl`) [30]. Since Wi-Fi does not expose an active-time counter, the granularity period is used as denominator. This is an approximation that holds under sustained load, where

Table 4.2: Wi-Fi to KPM metric mapping

KPM Metric Name	**Wi-Fi Source**
`DRB.UEThpDl`	Δ`tx_bytes` $\times$ 8 / period [bps]
`DRB.UEThpUl`	Δ`rx_bytes` $\times$ 8 / period [bps]
`DRB.RlcSduTransmittedVolumeDL`	Δ`tx_bytes` $\times$ 8 / 1000 [kbit]
`DRB.RlcSduTransmittedVolumeUL`	Δ`rx_bytes` $\times$ 8 / 1000 [kbit]
`DRB.RlcPacketDropRateDLDist`	Δ`tx_failed` / (Δ`tx_packets` + Δ`tx_failed`) $\times$ 100 [%]
`DRB.RlcPacketLossRateULDist`	Δ`rx_drop_misc` / (Δ`rx_packets` + Δ`rx_drop_misc`) $\times$ 100 [%]

the link is continuously active. For `DRB.RlcSduTransmittedVolume*`, the specification measures volume at the RLC SDU boundary [30]; the Wi-Fi MAC-layer byte counters used here include 802.11 framing overhead, making them a systematic overestimate of the RLC SDU volume quantified at approximately 12% in Section 6.2.2. Since both `DRB.UEThpDl` and `DRB.RlcSduTransmittedVolumeDL` derive from the same `tx_bytes` counter (differing only in the scaling factor applied), the overhead factor measured via the throughput comparison applies equally to the volume metric. The drop and loss rate formulas follow the standard ratio definition in TS 28.552 [30] without approximation.

The N3IWF correlates Wi-Fi station data with NGAP UE contexts to produce per-UE metrics aligned with the KPM measurement names above.

Each mapping was chosen as the closest semantic match in TS 28.552 [30]. In the NR stack, `DRB.UEThpDl` and `DRB.UEThpUl` are measured at the PDCP/SDAP layer (see Figure 4.2), above RLC segmentation and MAC framing. In the non-3GPP path, per-station byte counters are available at two levels: the **802.11 MAC layer** (cumulative `tx_bytes` and `rx_bytes` per associated station) and the **IPsec tunnel layer** (per-Security Association counters, which operate above the MAC and below IP). IPsec-level counters would be semantically closer to PDCP, since they exclude 802.11 headers and MAC-layer retransmissions. However, they would require correlating per-SA counters with UE contexts through the N3IWF's internal IKE state, and handling multiple Child SAs per UE (one signaling SA plus one per PDU Session), tightly coupling the metric collection to the N3IWF internals. The MAC-layer counters were therefore chosen: they are accessible from an external process without any dependency on the N3IWF. The trade-off is that MAC counters include 802.11 framing overhead and retransmitted frames, so KPM throughput values overestimate application-layer goodput by a systematic margin; Section 6.2.2 quantifies this offset. The `tx_failed` counter

tallies frames that exhausted their retry limit, analogous to RLC SDU drops (`DRB.RlcPacketDropRateDLDist`), while `rx_drop_misc` counts receive drops, analogous to uplink loss (`DRB.RlcPacketLossRateULDist`).

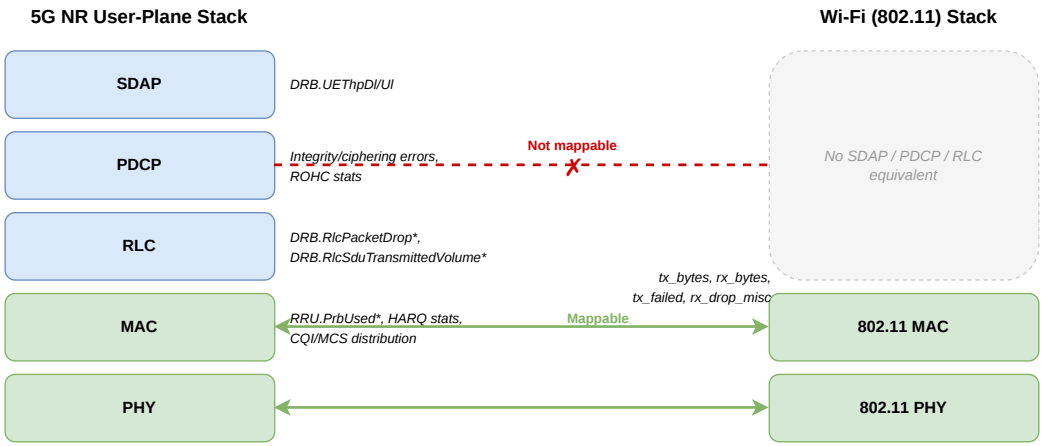

Figure 4.2: Protocol stack comparison between 5G NR (user-plane) and Wi-Fi, with the E2SM-KPM metric families that originate at each NR layer. Only MAC-level counters have a semantic counterpart in 802.11.

The mapping above covers only the subset of E2SM-KPM measurement names [30] for which a meaningful Wi-Fi counterpart exists. Figure 4.2 illustrates why. The 5G NR user-plane stack comprises five layers (PHY, MAC, RLC, PDCP, and SDAP), each exposing its own family of KPM metrics. Wi-Fi, by contrast, exposes station counters only at the MAC layer; there are no equivalents of RLC, PDCP, or SDAP.

Concretely, the following metric families cannot be populated from Wi-Fi:

- **PHY/MAC-layer scheduling metrics**: `RRU.PrbUsedDl`, `RRU.PrbUsedUl`, `RRU.PrbAvailDl`, `RRU.PrbAvailUl` (PRB utilization). Wi-Fi uses contention-based channel access (CSMA/CA) rather than scheduled resource blocks; no equivalent of per-slot PRB allocation exists. HARQ retransmission and NACK rates are likewise undefined, as Wi-Fi relies on frame-level ACK/retransmit without a HARQ process. CQI and per-subframe MCS distribution have no counterpart either, although Wi-Fi drivers report a current TX/RX MCS index per station, it is a single value rather than a distribution over scheduling intervals.

- **RLC-layer metrics**: packet-level loss semantics differ because Wi-Fi has no segmentation/reassembly layer comparable to RLC. The

`DRB.RlcSduTransmittedVolume*` metrics can be approximated by `tx_bytes`/`rx_bytes` (Table 4.2), but this approximation conflates MAC-layer retransmissions with actual data volume.

- **PDCP/SDAP-layer metrics**: integrity-protection error counters and ROHC header-compression statistics have no Wi-Fi equivalent. Wi-Fi encryption (CCMP/GCMP) is handled within the MAC layer; there is no separate ciphering/integrity layer above it.

This asymmetry is inherent: the E2SM-KPM measurement catalogue was designed for the NR protocol stack, and non-3GPP access technologies can populate only the metrics with a meaningful semantic counterpart at their own link layer.

4.4.3 End-to-End Telemetry Pipeline

The telemetry path from a Wi-Fi station to an xApp traverses three stages (Figure 4.3):

1. **Wi-Fi metric collection.** A lightweight exporter process co-located with the N3IWF periodically polls link-layer station counters (cumulative bytes, packets, errors) and produces structured per-station records.

2. **UE context correlation at the N3IWF.** The N3IWF correlates each Wi-Fi station with its NGAP UE context using its internal IKE/IPsec state, producing a unified per-UE record that associates 5G identity information (AMF UE NGAP ID, PLMN, S-NSSAI) with the station's link-layer counters.

3. **KPM encoding at the E2 agent.** The E2 agent reads the N3IWF's correlated per-UE records, computes rate-based metrics from the cumulative counters (see Table 4.2), and encodes the result as an E2SM-KPM Indication Message in Format 3 (one `MeasurementRecord` per UE). The Indication is delivered to the Near-RT RIC over the E2AP SCTP association, where it is forwarded to every xApp that holds an active KPM subscription for this E2 Node.

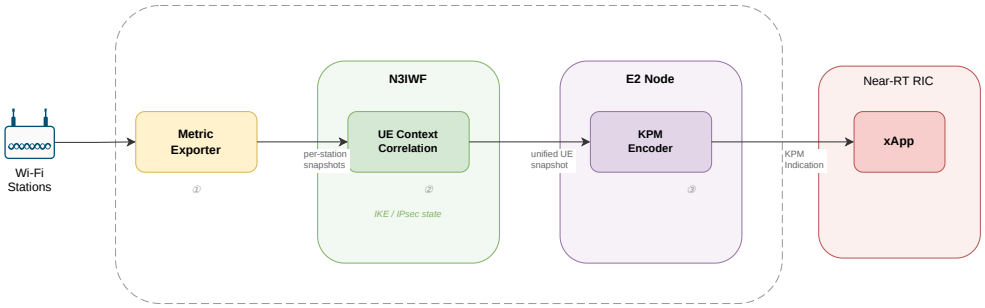

Figure 4.3: End-to-end telemetry pipeline. The Metric Exporter polls link-layer station counters from the Wi-Fi Access Point; the N3IWF correlates each station with its NGAP UE context; the KPM Encoder produces an E2SM-KPM Indication delivered to the xApp inside the Near-RT RIC.

This three-stage pipeline decouples metric collection (access-technology-specific) from metric delivery (O-RAN-standardized). Replacing the Wi-Fi exporter with a different collector (e.g., one that reads counters from a cellular small cell) would require no changes to the E2 agent or the xApp, provided the collector produces the same data schema.

4.4.4 Generalizability to Other Non-3GPP Access Technologies

The monitoring architecture described above is not specific to Wi-Fi. Of the three pipeline stages, only the first, the metric collector, depends on the underlying access technology: it must know how to query the link-layer counters of the particular medium. The remaining two stages are access-agnostic:

- The **N3IWF correlation step** matches station identifiers to NGAP UE contexts through IKE/IPsec state that is common to all non-3GPP access types traversing the N3IWF, regardless of the link layer underneath.
- The **E2 agent encoding step** reads a snapshot whose schema is defined in terms of cumulative byte, packet, and error counters per UE. Any collector that populates these fields enables the E2 agent to produce valid E2SM-KPM Indications without modification.

Likewise, the E2 Node identity (`GlobalE2node-gNB-ID`) is constructed from the N3IWF's PLMN and node ID, both core-network parameters that

are independent of the access technology. The RAN Functions registered at E2 Setup (KPM report styles, RC control styles) are also statically defined and carry no access-technology-specific content. The Near-RT RIC therefore sees a standard E2 Node regardless of whether the underlying link is Wi-Fi, satellite, or any other IP-capable medium.

The **handover execution mechanism**, however, does not generalize in the same way. The N2-based handover procedure described in Section 4.5 is specific to Wi-Fi access: the custom IKEv2 Notify payload carries Wi-Fi parameters (target SSID, AP address), and the execution flow assumes that the UE performs a Wi-Fi disassociation/re-association before the MOBIKE address update. While the IPsec state synchronization and the AMF-side N2 signaling are access-agnostic, adapting the execution phase to a different non-3GPP technology (e.g., satellite link switching) would require redesigning the UE triggering mechanism and the layer-2 handoff procedure. The E2SM-RC control interface and the agent-to-N3IWF communication remain unchanged: the xApp issues the same Control Request regardless of the access technology; only the N3IWF-to-UE execution path would need adaptation.

4.5 Control: N2-Based Handover via E2SM-RC

This section describes the N2-based handover mechanism between N3IWF instances, which realizes the actuation arm of the E2 closed loop through the E2SM-RC control interface. The xApp triggers the handover by sending an RC Control Request; the N3IWF then executes the adapted N2 procedure.

4.5.1 Choice of E2SM-RC Control Style 3

As discussed in Section 2.5.3, E2SM-RC defines multiple control styles for different classes of RAN actions. In this thesis, Control Style 3 ("Connected Mode Mobility Control") was selected because it directly maps to handover triggering. The control action ID is set to 1 ("Handover Control"), and the control message carries the target cell identity (encoded as NR CGI containing MCC, MNC, and N3IWF ID) that allows the E2 agent to identify the destination N3IWF.

4.5.2 N2 Handover Adaptation for Non-3GPP Access

The standard Inter NG-RAN node N2-based handover procedure [1] (described in Chapter 2) was designed for mobility between 3GPP RAN nodes

(gNBs). The rationale for choosing the N2 adaptation over Xn-based handover and session re-establishment is discussed in Section 4.3.2. This section details the concrete mechanism.

Access Topology: One AP per N3IWF

From the 5G Core's perspective, the N3IWF acts as the RAN node: it terminates the N2 interface towards the AMF and the N3 interface towards the UPF, analogously to a gNB. The Wi-Fi access points behind the N3IWF can be considered the equivalent of radio cells: intra-N3IWF AP mobility (multiple APs served by the same N3IWF) would be transparent to the core, much like intra-gNB cell handover, whereas inter-N3IWF mobility requires an N2-based handover involving the AMF.

In this work, a **one-to-one mapping** between APs and N3IWF instances is adopted. Each N3IWF is associated with exactly one Wi-Fi access point. This is a deliberate simplification that represents the *worst-case* scenario for handover cost: no AP change can be absorbed locally, and every mobility event requires the entire handover procedure (IPsec state transfer, path switch, downlink buffering, and MOBIKE exchange). If the mechanism performs adequately under this constraint, it will operate at least as well in a deployment where multiple APs share a single N3IWF and only a subset of AP switches require an inter-N3IWF handover.

Role of MOBIKE in the chosen approach As discussed in Section 2.3.2, MOBIKE (RFC 4555) [27] allows a UE to change its outer IP address without re-establishing the IKE SA. However, MOBIKE alone cannot perform inter-N3IWF mobility: it handles neither the transfer of UE context (NGAP state, IPsec SAs, PDU Session bindings) nor the update of N3 GTP-U tunnel endpoints at the UPF. In the chosen N2-based approach (Figures 4.4 and 4.5), MOBIKE is used as the *execution mechanism*: after the core network has prepared the target N3IWF and the IPsec state has been synchronized, the UE switches Wi-Fi association and performs a MOBIKE `UPDATE_SA_ADDRESSES` exchange with the target N3IWF to update the outer tunnel endpoint IP, avoiding a full IKEv2 re-negotiation and enabling the target to finalize the IPsec outbound rules with the UE's actual address. The UE-side execution steps are detailed in Section 4.5.2. The target then flushes the downlink buffer and sends the *Handover Notify* to the AMF.

Key Differences from Standard Inter-gNB N2 Handover

While the overall N2 handover signaling flow is preserved, the non-3GPP context introduces the following differences:

- **No RRC layer**: In 3GPP access, the S-RAN sends an RRC Reconfiguration to the UE, which synchronizes to the target cell. In N3IWF-based handover, there is no RRC. Instead, the target N3IWF pre-imports the full IPsec state (IKE SA and Child SAs) from the source via the Source-to-Target Transparent Container, and the source N3IWF signals the UE to begin the handover execution via a custom IKEv2 Notify mechanism (detailed in Section 4.5.2).
- **IPsec state transfer**: The standard handover transfers RRC/PDCP state in the Source-to-Target Transparent Container. In the N3IWF scenario, the equivalent is the transfer of IKE SA and Child SA state (cryptographic keys, SPIs, sequence numbers, traffic selectors) from the source N3IWF to the target N3IWF.
- **No radio resource reservation**: A gNB reserves radio bearers for the UE during preparation. The target N3IWF instead prepares IPsec tunnel parameters and allocates inner IP addresses. Wi-Fi radio resources are managed by the access point, not by the N3IWF.
- **Downlink buffering**: During a standard handover, the source gNB can buffer and forward downlink data over Xn or indirect forwarding tunnels. In the N3IWF case, the *target* N3IWF buffers downlink GTP-U packets arriving after the early path switch and flushes them once the UE completes the MOBIKE exchange. The rationale for the early path switch and target-side buffering is discussed in Section 4.3.2.

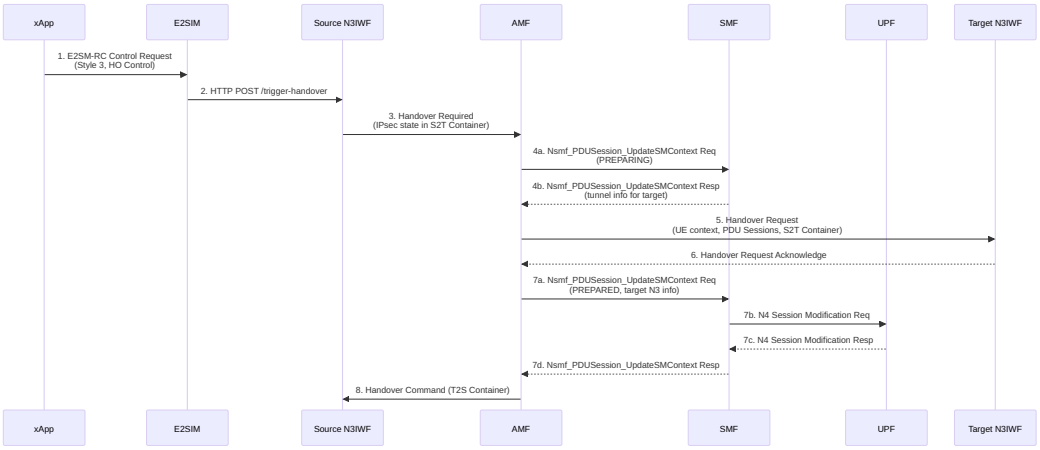

Figure 4.4: Adapted N2 handover preparation phase for non-3GPP access via N3IWF. The early path switch at the UPF (step 7b) occurs only after the target N3IWF has acknowledged the Handover Request.

Figures 4.4 and 4.5 show the adapted N2 handover procedure for non-3GPP access via N3IWF. The preparation phase (Figure 4.4) is triggered by the xApp via an E2SM-RC Control Request and follows the standard NGAP handover flow, with the key difference that the Source-to-Target Transparent Container carries serialized IPsec state instead of RRC context. The AMF first contacts the SMF in `PREPARING` state (steps 4a–4b) to obtain tunnel information for the target, then forwards the Handover Request to the target N3IWF (step 5). Only after the target acknowledges (step 6) does the AMF invoke the SMF again in `PREPARED` state (steps 7a–7c), at which point the SMF instructs the UPF to perform the early path switch. This ordering ensures that the UPF data path is never modified unless the target has confirmed its readiness.

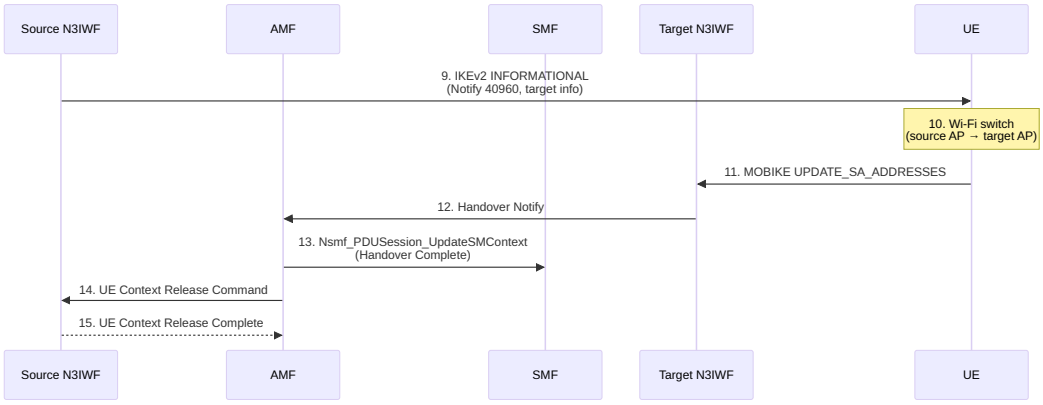

Figure 4.5: Adapted N2 handover execution phase for non-3GPP access via N3IWF (MOBIKE path). The source N3IWF signals the UE via a custom IKEv2 Notify, after which the UE switches Wi-Fi and completes MOBIKE with the target.

The execution phase (Figure 4.5) begins when the source N3IWF signals the UE via the custom IKEv2 Notify (step 9, see Section 4.5.2). The UE switches Wi-Fi association to the target AP and completes a MOBIKE `UPDATE_SA_ADDRESSES` exchange with the target N3IWF (step 11). The target then sends a Handover Notify to the AMF (step 12), which notifies the SMF of the handover completion (step 13) and releases the UE context on the source N3IWF (steps 14–15). In the MOBIKE path, no explicit NAS Registration Request is sent by the UE; the NWuCP (NAS over TCP) connection is re-established internally towards the target N3IWF after the MOBIKE exchange.

IPsec State Synchronization

A critical design element is the transfer of IPsec security association state from the source N3IWF to the target N3IWF. The state that must be transferred includes:

- **IKE SA state**: Cryptographic keys, SPIs, message IDs, and negotiated algorithms.
- **Child SA state**: For each Child SA (signaling SA + per-PDU-Session user-plane SAs): inbound/outbound SPIs, encryption and integrity keys, traffic selectors, ESN (Extended Sequence Numbers) state, and the associated QFI/PDU Session mappings.

The transfer is implemented by serializing this state into a structured payload and transmitting it within the Source-to-Target Transparent Container of the NGAP Handover Required message. The target N3IWF deserializes the state and installs the corresponding IPsec policies and security associations, effectively "importing" the IPsec tunnel without requiring a new IKEv2 negotiation with the UE.

The security of this transfer relies on the N2 interface being protected by NDS/IP (Network Domain Security for IP) [23], which mandates IPsec-based protection of inter-NF communication in production deployments. In the testbed, the SCTP link between N3IWF and AMF is unprotected; the serialized cryptographic material therefore traverses the control plane in cleartext, which is acceptable for a laboratory environment but would require NDS/IP enforcement in any real deployment.

Downlink Buffering Strategy

As discussed in Section 4.3.2, an early path switch is used: the UPF redirects the downlink N3 GTP-U tunnel to the target N3IWF during the preparation phase, before the UE has begun the MOBIKE exchange. However, the target N3IWF has imported the IPsec state from the source, so its outbound Security Association rules still contain the source's peer outer IP address. Until the UE completes the MOBIKE `UPDATE_SA_ADDRESSES` exchange (which updates the outbound rules with the UE's real outer IP on the target's Wi-Fi network), any downlink packet written to the IPsec tunnel would be sent to the wrong destination. To prevent packet loss during this window, the *target* N3IWF activates a per-TEID (Tunnel Endpoint Identifier) downlink buffer. The buffer is activated as soon as the GTP-U tunnels are established; buffered packets are flushed to the UE immediately after MOBIKE completes and the IPsec rules are updated.

UE-Side Execution: Wi-Fi Switch and MOBIKE

While the handover preparation is entirely network-side (source N3IWF, AMF, SMF, UPF, target N3IWF), the execution phase requires active participation from the UE. The source N3IWF must signal the UE to begin the handover execution after the preparation phase is complete.

Signaling the UE: IKEv2 Notify as the non-3GPP equivalent of RRC In 3GPP access, the gNB triggers UE-side handover execution by sending an RRC Reconfiguration message, a direct signaling channel between the RAN node and the UE. In the non-3GPP context, no such channel exists

natively. The only direct signaling channel between the N3IWF and the UE is the IKEv2 Security Association, which is already established, encrypted, and integrity-protected.

The source N3IWF triggers the handover execution by sending an IKEv2 `INFORMATIONAL` exchange containing a Notify payload with a private-use message type (within the range defined by RFC 7296 [26]). The payload carries the target N3IWF's connection parameters and the SSID of the target Wi-Fi access point. This serves the same triggering role as the RRC Reconfiguration in 3GPP access (signaling the UE to begin the handover execution), but is a custom protocol extension that requires matching support on both the N3IWF and the UE.

Execution steps Upon receiving the IKEv2 handover notification, the UE:

1. Disassociates from the source AP and associates with the target AP (Wi-Fi switch). The Wi-Fi switch duration depends on the 802.11 association procedure and the behavior of the Wi-Fi driver and supplicant, and is expected to be a significant contributor to the total handover latency.

2. Sends a MOBIKE `UPDATE_SA_ADDRESSES` request to the target N3IWF using its new outer IP address on the target Wi-Fi network.

3. The target N3IWF accepts the MOBIKE exchange, updates the IPsec outbound rules with the UE's new peer IP, flushes the downlink buffer, and sends *Handover Notify* to the AMF.

After the Handover Notify, the UE re-establishes the NWuCP (NAS over TCP) connection towards the target N3IWF's inner IP address within the IPsec tunnel. No explicit NAS *Registration Request* (Mobility Registration Update) is sent: the IPsec state synchronization ensures that the NAS security context is already available at the target N3IWF, and the AMF has already updated the UE's serving RAN node through the Handover Notify.

4.6 xApp Design

The Handover xApp implements a load-balancing strategy that redistributes UEs across N3IWF instances based on periodic KPM telemetry.

4.6.1 Load-Balancing Strategy

The xApp follows a threshold-based, window-aggregated approach:

1. The xApp subscribes to E2SM-KPM reports from *all* connected E2 Nodes (N3IWFs).

2. For each E2 Node, a report counter is maintained. Each incoming KPM Indication increments the counter for the corresponding node.

3. When *any* node has accumulated a fixed number of KPM Indications, the xApp evaluates the load distribution across all subscribed nodes.

4. The load metric is the number of UEs reported in the KPM Indication Message (i.e., the length of the `ueMeasReportList` in Format 3).

5. The node with the highest UE count is selected as the *source*, and the node with the lowest UE count as the *target*.

6. If a load imbalance exists (source UE count $>$ target UE count), the xApp selects the *first* UE in the source node's most recent KPM report list and sends an E2SM-RC Control Request (Style 3, Action ID 1) to trigger an N2 handover to the target. This simple selection policy is sufficient for the proof-of-concept testbed; a production xApp could incorporate per-UE metrics (e.g., throughput, signal quality) to make a more informed choice.

7. All counters are reset after the decision, and the cycle restarts.

The xApp operates as a single observe-decide-act loop without persistent policy installation. Each handover is triggered as a one-shot RIC Control Request, and no concurrent policies are maintained. This simplifies the design and avoids conflicts between overlapping control decisions.

4.6.2 Control Loop Timing

The xApp's decision cycle is governed by two parameters: the KPM *reportingPeriod* (the interval at which the E2 agent transmits one Indication message) and the report *threshold* (the number of Indications accumulated before a load evaluation is triggered, set 20 in this work). Together, these parameters determine the minimum time between two consecutive handover decisions: at least $20 \times 1\,\mathrm{s} = 20\,\mathrm{s}$ must elapse after each decision before the next one can be taken.

This 20-second window provides a deliberate settling period: after a handover, the target N3IWF needs several KPM reporting cycles before its telemetry reflects the post-handover load (the migrated UE must appear in the target's reports and throughput must stabilize on the new path). A threshold that is too low would risk triggering consecutive handovers before the metrics have stabilized, potentially oscillating the UE between two N3IWFs.

Once the threshold is reached and a handover is decided, the xApp sends the RIC Control Request and receives an immediate RIC Control Acknowledge (or Failure) from the E2 agent. Regardless of the outcome, all report counters are reset, so the next decision opportunity arises only after a full threshold window of fresh KPM reports. If the handover fails, no retry is performed: the cycle restarts from scratch. This avoids compounding failures with immediate retries.

4.7 Design Limitations

The sidecar architecture and the handover mechanism described above involve deliberate trade-offs whose implications should be understood when evaluating the design or planning extensions.

- **Co-located deployment.** The E2 agent and the N3IWF must run on the same host because both communication channels between them (the shared metrics file consumed by the KPM worker and the configuration file read at E2 Setup) rely on a shared filesystem. This co-location requirement means the agent cannot be deployed remotely: each N3IWF instance requires its own dedicated co-located agent. In a small testbed with two N3IWF nodes this is not a concern, but in a large-scale non-3GPP deployment it would preclude any consolidation of the E2 layer. Replacing the shared-file channels with a networked alternative (e.g., a gRPC stream or a message queue) would lift the co-location constraint and would in principle allow a centralised agent to manage multiple N3IWF instances over separate E2 connections, at the cost of added complexity and an additional network hop on the control path.

- **Polling-based metric delivery.** The E2 agent reads the N3IWF's metric state at the KPM *granularityPeriod* interval; there is no push notification from the N3IWF. For periodic KPM reports this is a natural fit, since the agent must produce one Indication per period regardless of when the data changed. However, it would be suboptimal for

event-triggered E2SM indications (e.g., threshold-crossing alerts): an event occurring between two polls would not be visible until the next read. An event-driven channel would reduce this latency.

- **Static metric schema coupling.** The set of metrics flowing from the N3IWF to the E2 agent is defined by a fixed schema. Adding a new KPM metric requires updating both the N3IWF (to emit the new counter) and the agent (to parse and encode it); there is no dynamic discovery mechanism. This is acceptable for the current six-metric set but would become a maintenance concern as the number of supported metrics grows. A self-describing format (where the payload advertises the metric names it carries) would decouple the two components at the cost of a slightly more complex parser.

- **Intra-AMF constraint.** The current design is limited to handover between N3IWF instances registered with the same AMF. The inter-AMF UE context transfer procedure [1] (steps 3–4 of the standard preparation phase described in Section 2.2.6) is not addressed. This simplification removes the need for AMF relocation but does not affect the core handover mechanism, which would remain unchanged once inter-AMF context transfer is supported.

Chapter 5

O-RAN Closed-Loop Control over Non-3GPP Access: System Implementation

This chapter details the implementation of the system designed in Chapter 4. It begins with the E2 Node integration (E2SIM agent and ASN.1 codec layer), proceeds to the Handover xApp, and then describes the N3IWF handover logic and the modifications required in the free5GC AMF and SMF.

The N3IWF[1], AMF[2], and SMF[3] are based on the free5GC [16] open-source project and are implemented in Go. The E2SIM agent is implemented in C/C++ and communicates with the N3IWF through two channels: shared files written by the N3IWF (read by the E2SIM for KPM metrics and configuration) and HTTP POST requests sent by the E2SIM to the N3IWF (for RC control actions). The xApp is implemented in Python using the xDevSM framework. Figure 5.1 shows the complete container deployment supporting both service models.[4]

[1] `https://github.com/Zhria/n3iwf`
[2] `https://github.com/Zhria/amf`
[3] `https://github.com/Zhria/smf`
[4] Full deployment configuration: `https://github.com/Zhria/free5gc-compose`.

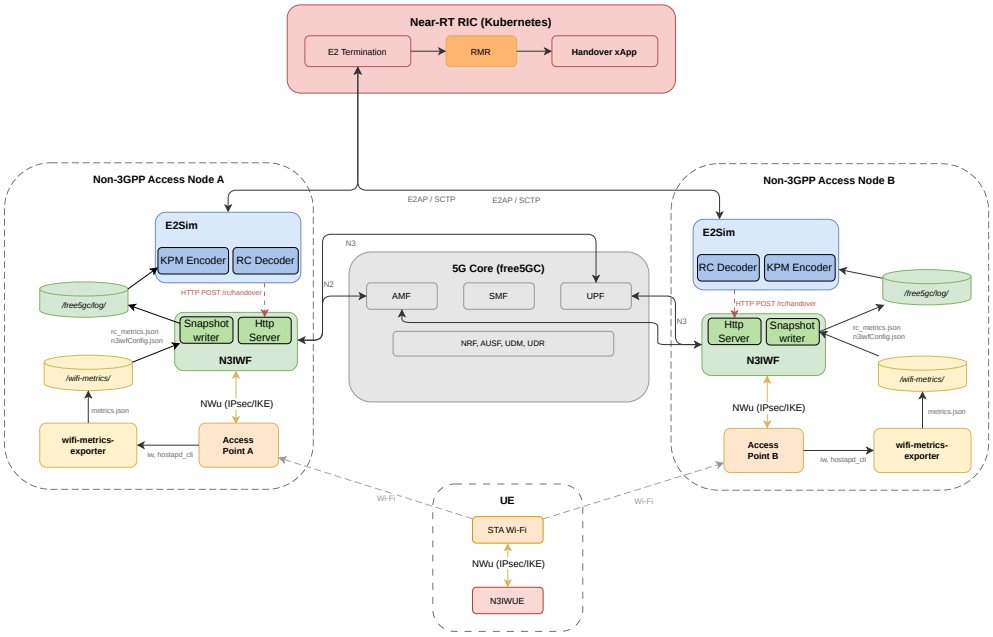

Figure 5.1: Container deployment supporting both E2SM-KPM and E2SM-RC service models. Each Non-3GPP Access Node runs three containers (wifi-metrics-exporter, N3IWF, E2SIM) connected via shared Docker volumes. The E2SIM communicates with the Near-RT RIC over E2AP/SCTP and triggers handovers on the N3IWF via HTTP.

5.1 E2SIM Integration

The E2SIM[5] agent runs as a separate process alongside each N3IWF instance. It implements the E2AP protocol (Section 2.5) over SCTP and exposes two RAN Functions: E2SM-KPM (ID=2, Section 2.5.2) and E2SM-RC (ID=3, Section 2.5.3).

The starting point was the OSC e2sim [50], a reference E2 Node stub that generated hardcoded synthetic metrics with no mechanism to ingest data from a real network element. It supported only E2AP v0.31 and E2SM-KPM v1; E2SM-RC was not implemented. To turn it into a functional agent for the N3IWF, the encoding layer was rewritten for the current O-RAN specifications (E2AP v3.01, E2SM-KPM v3, E2SM-RC v1.03), RC control-request handling was built from scratch, and a data-ingestion pipeline was created so that the agent reads live Wi-Fi and UE state from the N3IWF at

[5] `https://github.com/Zhria/e2sim`

runtime rather than producing static values.

5.1.1 ASN.1 Code Generation

To support E2AP v3.01, E2SM-KPM v3.00, and E2SM-RC v1.03, the entire ASN.1 codec layer was regenerated.

Source definitions. Three ASN.1 files in `asnfiles/` provide the type definitions retrieved from the flexric project [51]:

- `e2ap_v3_01.asn`: E2AP v3.01 protocol definitions.
- `e2sm_kpm_v03.00_modified.asn1`: E2SM-KPM v3.00 Information Element definitions, adapted from the official O-RAN ASN.1 to resolve compilation issues (e.g., parameterized type workarounds, import adjustments).
- `e2sm_rc_v1_03_modified.asn`: E2SM-RC v1.03 Information Element definitions, also adapted for compiler compatibility.

Compiler. The C codec stubs were generated using the `mouse07410/asn1c` fork [52], which is the ASN.1 compiler recommended by the O-RAN SC Requirements and Software Architecture Committee (RSAC) since March 2025 [53]. The compiler was invoked with the following flags:

```
asn1c -gen-PER -fcompound-names -fno-include-deps \
      -no-gen-OER -no-gen-example -pdu=all \
      -D ASN1c/ asnfiles/*.asn*
```

where `-gen-PER` generates Aligned PER (APER) codecs as required by E2AP, `-fcompound-names` avoids name collisions across ASN.1 modules, and `-no-gen-OER` excludes unused OER codecs.

Output and build integration. The CMake build globs all generated `.c` files and filters out unused codec helpers (`ber_`, `der_`, `jer_`, `oer_`, `uper_`) to reduce the binary size. The ASN.1 upgrade required rewriting the entire encoding layer: `encode_e2apv1.cpp` was replaced by `encode_e2apv3.cpp` (E2AP v3 message construction), `encode_kpm.cpp` was rewritten for KPM v3, and `encode_rc.cpp` was created from scratch for RC v1.03.

5.1.2 E2 Setup and RAN Function Registration

At startup, the E2SIM:

1. Reads configuration from `e2node.yaml` (E2 Termination address, gNB-CU-UP-ID, gNB-DU-ID). The latter two are placeholder values required by the Near-RT RIC's E2 Manager node registration logic to suppress spurious error logs; they carry no semantic meaning for a non-3GPP node.

2. Reads `n3iwfConfig.json`, a JSON file that the N3IWF overwrites every second with its full configuration and runtime UE context. The E2SIM extracts the PLMN ID and N3IWF ID from this file.

3. Establishes an SCTP association to the E2 Termination in the Near-RT RIC.

4. Sends an *E2 Setup Request* advertising a `GlobalE2node-ID` (gNB type with N3IWF ID as gNB-ID) and two RAN Functions:

   - **ID=2, E2SM-KPM v3**: with EventTrigger Style 1 (periodic), Report Style 4 (common condition-based, UE-level), and the supported metric list from `getAllowedKPI()`.
   - **ID=3, E2SM-RC v1.03**: with EventTrigger Style 4 (UE Information Change), Control Style 3 (Connected Mode Mobility Control), and 4 control parameters.

5.1.3 E2SIM–N3IWF Communication

The E2SIM (C/C++) and the N3IWF (Go) run in separate Docker containers on the same host, communicating through the two unidirectional channels described in Section 4.2.1. Figure 5.2 shows the internal structure of the E2 Node, detailing the functional components and the inter-process communication channels. This section details the concrete implementation of each channel.

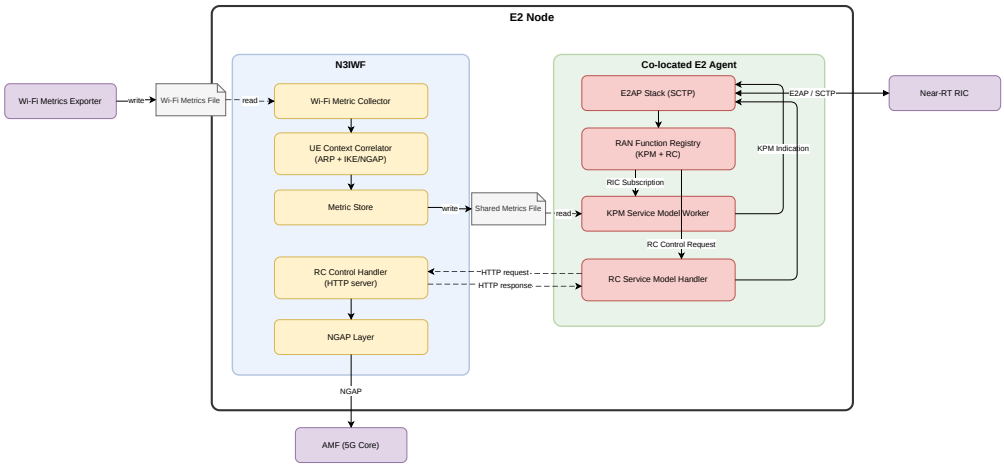

Figure 5.2: Internal structure of the E2 Node: functional components of the N3IWF and the co-located E2 Agent, with inter-process communication channels (shared metrics file and HTTP control path).

Observation path (N3IWF → E2SIM). The N3IWF writes two JSON files to a shared Docker volume (`/free5gc/log/`):

- `n3iwfConfig.json`: overwritten every second with the N3IWF's full runtime state.
- `rc_metrics.json`: overwritten every 500 ms with per-UE association records (see Appendix A for the full JSON structure).

Both files are overwritten in-place on the Go side via a truncate-and-rewrite pattern (`Truncate(0)` + `Seek(0,0)` + `Write`). On the C++ side, the E2SIM parses each file using the `nlohmann/json` library.

Control path (E2SIM → N3IWF). The E2SIM sends RC control commands via HTTP POST to the N3IWF's `/rc/handover` endpoint, using `libcurl`. The endpoint URL is configurable via the `n3iwfHandoverUrl` field in `e2node.yaml` or the `RC_HANDOVER_TRIGGER_URL` environment variable. The request uses a 950 ms client-side timeout to remain within the Near-RT RIC's control-loop budget; the call blocks until the N3IWF responds or the timeout fires (Section 5.3.1).

5.1.4 KPM Implementation

Wi-Fi Metrics Collection Pipeline. The KPM data pipeline consists of three stages:

1. **Wi-Fi exporter** (`exporter.py`): A Python script deployed as a dedicated Docker container (`wifi-metrics-exporter`) alongside each N3IWF. The container runs with `privileged: true` and `pid: host` to access the host's Wi-Fi interfaces directly. Every 500 ms (configurable via `SCRAPE_INTERVAL`), it collects per-station data from two sources: `iw dev <iface> station dump` for kernel-level counters (bytes, packets, signal, bitrate) and `hostapd_cli all_sta` for association-level information (connected time, capabilities). The output is written atomically to a shared Docker volume (`/wifi-metrics/`) mounted by both the exporter and the N3IWF container.

2. **N3IWF correlation**: Every 500 ms, the N3IWF reads the exporter's JSON output and correlates each Wi-Fi station with the corresponding NGAP UE context through a three-step lookup: MAC address → IP address (via ARP table), then IP address → IKE SA (by matching the station's IP against the inner IPsec addresses of connected UEs), and finally IKE SA → NGAP context (via the `IKESPIToNGAPId` mapping maintained in the N3IWF's runtime snapshot). The resulting per-UE records, combining 5G identity (AMF UE NGAP ID, PLMN, S-NSSAI) with Wi-Fi counters, are written to `rc_metrics.json` (see Appendix A for the full JSON structure).

3. **E2SIM KPM worker**: A dedicated thread in the E2SIM reads `rc_metrics.json` every `granularityPeriod` (the aggregation window from the subscription's Action Definition), computes delta values between consecutive snapshots, and accumulates the results. Once enough collection cycles have elapsed to cover one `reportingPeriod` (from the subscription's Event Trigger), the worker assembles all accumulated measurements into a single E2SM-KPM Indication Message (Table 4.2) and sends it to the Near-RT RIC.

Subscription lifecycle. Each active KPM subscription is tracked by a `SubscriptionKey` tuple (requestorId, instanceId, ranFunctionId). Upon receiving a *RIC Subscription Request*, the E2SIM validates the requested metrics against `getAllowedKPI()`, sends a *RIC Subscription Response*, and spawns a dedicated worker thread. If a subscription with the same key already exists, the previous worker is stopped (via atomic flag) and joined before the new one starts. Multiple concurrent subscriptions from different xApps are supported; each worker operates independently on its own timer and metric set.

Report loop. The worker extracts the `reportingPeriod` from the subscription's Event Trigger and the `granularityPeriod` from the Action Definition. Each reporting cycle proceeds as follows:

1. **Accumulation**: the worker repeatedly sleeps for one `granularityPeriod`, reads `rc_metrics.json`, and calls `getMetricsKPMByRanUeId()`, which computes delta values between consecutive snapshots (handling counter wraparound via unsigned subtraction) and normalizes throughput metrics by the elapsed time in seconds. This continues until one full `reportingPeriod` has been covered.

2. **Indication assembly**: the worker builds an `E2SM-KPM-IndicationMessage` in Format 3: one `UEMeasurementReportItem` per UE, each wrapping a Format 1 record with the metric values requested by the subscription.

3. **Validation**: the message is APER-encoded, then a self-decode check is performed: the encoded bytes are immediately decoded back into an ASN.1 structure, and each UE entry is validated (non-null, non-empty `measData`). If the self-decode fails, the report is silently dropped rather than delivering a corrupt indication.

4. **Transmission**: the encoded header and message are wrapped into an E2AP *RIC Indication* PDU and sent over SCTP.

5.1.5 RC Implementation

Control Request Handling. When the E2SIM receives a *RIC Control Request* (Style 3, Action ID 1), the callback `callback_rc_control_request()` processes it in three stages:

Header decoding. The E2SM-RC Control Header (Format 1) is APER-decoded to extract the `UEID`, from which the `RAN_UE_NGAP_ID` identifies the target UE. The style and action ID are also validated against the supported control style.

Message decoding. The E2SM-RC Control Message (Format 1) contains a flat list of `RANParameter` items. The decoder iterates the list and, for each recognized parameter ID, extracts the value. The target cell identity is obtained from the *Target Primary Cell ID* parameter (ID=4), which carries an NR CGI encoded as an 8-byte OCTET STRING: the first 3 bytes encode

the PLMN Identity in BCD format, and the remaining 5 bytes encode the 36-bit NR Cell Identity. The decoder extracts MCC, MNC, and NR Cell ID from these bytes and formats them as a target identifier string (e.g., `208-93-136`) that the N3IWF can interpret as a `GlobalN3IWFID`.

Execution and outcome. The function `execute_rc_control_command()` constructs a JSON payload containing the resolved `ranUeNgapId` and target ID, POSTs it to the N3IWF endpoint (Section 5.1.3), and maps the HTTP response to an E2SM-RC Control Outcome. The outcome is encoded with two parameters: *Execution Status* (ID=50001, success or failure) and *Execution Notes* (ID=50002, the N3IWF's textual response). If the HTTP call succeeds ($2xx$), a *RIC Control Acknowledge* is sent; otherwise, a *RIC Control Failure* with the appropriate `Cause` IE is returned.

5.2 Handover xApp Implementation

The Handover xApp[6] is implemented in Python using the xDevSM framework [31] (Section 2.6) and deployed as a Docker container on the Near-RT RIC platform.

5.2.1 Deployment

The xApp deployment follows the O-RAN xApp lifecycle [54]:

1. The xApp is packaged as a Docker image and pushed to Docker Hub.
2. An xApp descriptor (`config-file.json`) and schema file (`schema.json`) are prepared, declaring the container image, RMR messaging ports (including the set of transmitted and received E2AP message types), and the subscription endpoint URL.
3. The `dms_cli` tool, together with ChartMuseum and Helm, generates and deploys the xApp Helm chart on the Kubernetes-based Near-RT RIC platform (j-release).

[6] `https://github.com/Zhria/xDevSM-xapps-examples`

5.2.2 xDevSM Abstraction Layer

The xApp does not construct E2AP or E2SM PDUs manually. The xDevSM framework [31] provides two decorator classes that abstract the E2 interaction:

- **XappKpmFrame**: handles KPM subscription creation (via the Subscription Manager REST API), E2AP indication decoding, and APER decoding of E2SM-KPM messages through C shared libraries (`libkpm_sm.so` for KPM, `librc_1_03.so` for RC) loaded via `ctypes`. Incoming KPM Format 3 indications are decoded into structured Python objects that expose the per-UE measurement report list (e.g., `ind_msg.data.frm_3.meas_report_per_ue[i]`) with typed accessors for UE ID and measurement values. The xApp registers a callback (`ind_msg_handler`) that receives the already-decoded header and message objects.
- **ConnectedModeMobilityControl**: a decorator for RC Control Style 3. When the xApp decides to trigger a handover, it configures the target cell via `set_plmn_identity()` and `set_nr_cell_id()` (retrieved from the E2 Manager's node information), then calls `send()` with the UE ID structure selected by the control logic from the KPM indications received for the source E2 Node. The decorator internally generates the E2SM-RC Control Header (Format 1, embedding the UE ID) and Control Message (Format 1, with the NR CGI parameter), APER-encodes both via the C shared libraries, and dispatches the RIC Control Request via RMR. The xApp is entirely decoupled from ASN.1 encoding.

5.2.3 Subscription to E2 Nodes

At startup, the xApp:

1. Queries the Near-RT RIC platform for the list of connected E2 Nodes (gNBs/N3IWFs).
2. For each node exposing both KPM and RC RAN Functions, sends a KPM subscription request for periodic reports via the `XappKpmFrame` subscription API.
3. Stores the RC RAN Function descriptions for later use in control requests.

5.2.4 Observe–Decide–Act Loop

The core xApp class (`xAppMonControlContainer`) implements three methods that realize the observe–decide–act loop (the full implementation is provided in Listing A.5, Appendix A).

Observe. The `ind_msg_handler` callback is invoked for each incoming KPM Indication. It increments the per-node report counter and, for Format 3 messages, extracts the UE count and caches the first UE ID struct (needed later as the handover candidate).

Decide. When any node's counter reaches the configurable threshold, the handler calls `_select_source_target`, which identifies the most-loaded and least-loaded E2 Nodes by comparing their latest UE counts. If fewer than two nodes are subscribed, no handover is possible.

Act. If a load imbalance exists, `_try_send_handover` retrieves the target node's PLMN ID and NR Cell ID from the E2 Manager's RAN info, configures the `ConnectedModeMobilityControl` decorator, and dispatches the RC Control Request (Style 3, Action ID 1) targeting the first UE in the source's report list. All counters are reset after the dispatch, and further handovers are suppressed until the Near-RT RIC platform delivers either a *RIC Control Acknowledge* or *RIC Control Failure*.

E2 Node identification. The xApp distinguishes E2 Nodes through their `inventory_name`, a string that the E2 Manager derives from the `GlobalE2node-ID` advertised in the E2 Setup Request. On the E2SIM side, the `GlobalgNB-ID` is constructed by reading the N3IWF's `N3IWFID` (mapped to the `gnb-ID` bit string) and `PLMN-ID` from `n3iwfConfig.json` (Section 5.1.2). When the xApp needs to send an RC Control Request to the source node and specify the target cell, it retrieves the target's `plmnId` and `nbId` from the `globalNbId` field in the E2 Manager's RAN info, which reflects the same `GlobalE2node-ID` the target E2SIM advertised at setup. This mechanism ensures that each N3IWF instance is uniquely identified across the Near-RT RIC platform, and the xApp can unambiguously select source and target nodes based on the per-node UE count reported via KPM.

5.3 N3IWF Handover Implementation

The subsections below follow the chronological order of the handover procedure: the RC trigger initiates the process, the source N3IWF builds the NGAP messages and serializes the IPsec state, the target responds with the Target-to-Source container, the UE executes the Wi-Fi switch and MOBIKE update, downlink packets are buffered until the data path is restored, and finally the source cleans up the old UE context. The section concludes with the network offload workaround required on the IPsec/data-plane path.

5.3.1 Handover Trigger via RC

The handover is triggered externally through an HTTP API exposed by the N3IWF (the RC Control Handler HTTP server visible in Figure 5.2). The implementation is in the `internal/rc/` package.

```
type HandoverAlert struct {
    RanUeNgapId          int64
    Cause                *ngapType.Cause
    TargetID             *ngapType.TargetID
    DirectForwarding     bool
    SourceToTargetContainer []byte
    Metadata             map[string]string
}

func StartHandoverHTTPServer(addr string)
```

Listing 5.1: Handover HTTP trigger and alert structure

The `StartHandoverHTTPServer` function launches a background HTTP server (default port 9085) that listens on the `POST /rc/handover` endpoint. The JSON payload contains the `ranUeNgapId` of the UE to be handed over and the `targetId` in the format `MCC-MNC-N3IWFID` (e.g., `"208-93-136"`). The handler:

1. Parses the target ID string and constructs an NGAP `TargetID` with `GlobalN3IWFID`.
2. Validates the UE state (existence, not already in handover, has active PDU Sessions, has associated AMF).
3. Dispatches a `TriggerHandoverEvt` to the NGAP handler, which starts the N2 handover preparation (Figure 4.4).

4. Responds immediately with `handover_triggered` once the NGAP event has been queued. Validation errors (unknown UE, invalid target) are returned as synchronous HTTP errors.

Because the N3IWF responds before the handover completes, the E2SIM can return a *RIC Control Acknowledge* to the Near-RT RIC within milliseconds. The actual handover (preparation + execution) proceeds asynchronously; the xApp monitors subsequent KPM reports to observe the outcome. The E2SIM still enforces a client-side HTTP timeout (see Section 5.1.3) as a safety net: if the N3IWF does not respond within that window (e.g., due to a validation stall), a *RIC Control Failure* is returned instead.

5.3.2 NGAP Handover Messages

The N3IWF implements the NGAP handover message construction functions in `internal/ngap/message/`. The messages themselves follow the N2 handover procedure described in Section 2.2.6; this section focuses on the non-obvious implementation choices that arise from applying the 3GPP handover framework to a non-3GPP access node.

RRCContainer reuse for IPsec state. The `BuildHandoverRequired` function serializes the full IPsec state (Section 5.3.3) into the `RRCContainer` field of the `SourceNGRANNodeToTargetNGRANNodeTransparentContainer`. In a gNB handover, this field carries RRC context; in the N3IWF case it carries JSON-encoded IKE/Child SA state. The AMF relays the container opaquely, so the reuse is transparent and avoids the need for a custom container format.

Failure and cancellation paths. The N3IWF implements `BuildHandoverPreparationFailure` (sent by the target on IPsec import failure or resource exhaustion), `BuildHandoverCancel` (sent by the source to abort a handover already in preparation), and two per-PDU-Session transfer helpers: `BuildHandoverRequestAcknowledgeTransfer` (carrying the DL GTP-U endpoint) and `BuildHandoverResourceAllocationUnsuccessfulTransfer` (carrying the rejection cause for sessions that could not be admitted).

5.3.3 IPsec State Synchronization

The IPsec state synchronization is implemented in the `internal/handover/statesync/` package. It provides serialization and deserialization of the full IKE/IPsec state required to reconstruct security associations on the target N3IWF. The role of this component within the overall handover flow is illustrated in Figure 4.4, where the serialized state is carried inside the Source-to-Target Transparent Container (step 3).

State Data Structures. The state transfer is modeled by two Go structs: `IKESAState` captures the IKE SA-level parameters (SPIs, message counters, negotiated transforms, and all derived session keys, NAT and MOBIKE flags), while `ChildSAState` captures per-Child-SA parameters including the ESP SPIs, SA role, traffic selectors, UDP encapsulation settings for NAT traversal, XFRM interface ID, negotiated cryptographic transforms, XFRM anti-replay counters, and four directional keys (initiator-to-responder and responder-to-initiator, for both encryption and integrity). The full struct definitions are provided in Listing A.4 (Appendix A).

State Import on the Target N3IWF. The target N3IWF's handover handler first APER-decodes the `SourceNGRANNodeToTargetNGRANNode-TransparentContainer`, extracts the JSON payload from the `RRCContainer` field, and deserializes it into a `StateTransfer` struct. The resulting struct is passed to `ImportStateForRanUe`, which:

1. Reconstructs the IKE SA object with all cryptographic contexts, incrementing the `ResponderMessageID` to account for the INFORMATIONAL exchange consumed during handover.
2. For each Child SA, installs the corresponding Linux XFRM state (inbound and outbound) and XFRM policy rules using the `netlink` library.
3. Creates the XFRM interface (`ipsec-N`) if it does not already exist.
4. Reserves the UE inner IP in the local pool and registers the IKE UE in the N3IWF context.

5.3.4 Target-to-Source Container and UE Notification

The Target-to-Source Transparent Container carries the information the UE needs to execute the handover (steps 6 and 9 in Figures 4.4 and 4.5, respec-

tively). It is built by the *target* N3IWF during `HandleHandoverRequest` and relayed to the source via the AMF's *Handover Command*.

Container construction. The function `buildTargetToSourceContainer` on the target N3IWF serializes a JSON object containing:

- **Access**: target N3IWF IP address, IKE port, and FQDN.
- **NAS**: NAS security context (NCC, NH, 5G-GUTI) for the UE to resume NAS signaling on the target.
- **Wi-Fi**: SSID and password of the target access point.
- **PDU Sessions**: per-session metadata (ID, UPF address, TEID, QFI list, GTP bind address).

This JSON is placed directly as the `TargetToSourceTransparentContainer` IE in the NGAP *Handover Request Acknowledge*.

Delivery to the UE. When the source N3IWF receives the *Handover Command* from the AMF, `HandleHandoverCommand` extracts the `TargetToSourceTransparentContainer` and stores it in the shared UE context. It then calls `sendTargetToSourceToUE`, which sends an IKEv2 `INFORMATIONAL` exchange containing a Notify payload with private-use type 40960. The UE receives the container and begins the execution phase (Wi-Fi switch + MOBIKE) as described below.

5.3.5 N3IWUE Client Modifications

The N3IWUE[7] is the non-3GPP UE client that executes the handover on the UE side. It is based on the free5GC n3iwue [55] and was extended to support the handover execution triggered by the IKEv2 Notify from the source N3IWF.

Handover execution. Upon receiving the IKEv2 Notify, the N3IWUE deserializes the JSON container and extracts the target N3IWF address, IKE port, and the target AP's SSID and credentials, while concurrently triggering a Wi-Fi rescan (Section 5.3.5). The Wi-Fi switch uses a two-tier strategy: first, `nmcli con up` attempts a fast reconnection using a saved

[7] `https://github.com/Zhria/n3iwue`

NetworkManager profile (no DHCP if a static IP is configured); if this fails (e.g., no saved profile exists), the UE falls back to `nmcli dev wifi connect` with the SSID and password received in the container. If both attempts fail, the UE rolls back to the source AP and aborts the handover. After a successful Wi-Fi switch, the UE sends a MOBIKE `UPDATE_SA_ADDRESSES` to the target N3IWF using its new outer IP. Once the target acknowledges, the UE re-establishes the NWuCP session (NAS over TCP) towards the target N3IWF's inner IP address within the IPsec tunnel, resuming NAS signaling without an explicit Mobility Registration Update.

Wi-Fi scan optimization. To minimize the Wi-Fi re-association latency during handover, the N3IWUE triggers an on-demand `nmcli device wifi rescan` immediately upon receiving the Target-to-Source Transparent Container (i.e., the handover command from the source N3IWF). A brief delay (300 ms) is introduced before the actual connection switch to allow NetworkManager's AP cache to be populated by the scan results. This avoids the full directed probe scan that `wpa_supplicant` would otherwise perform during `nmcli con up`, which was observed to add approximately 3–4 seconds to the handover latency.

MOBIKE fallback. The UE attempts a MOBIKE `UPDATE_SA_ADDRESSES` exchange (Section 2.3.2) as the primary handover execution mechanism. If MOBIKE fails, the UE falls back to a full IKE re-establishment towards the target N3IWF (which includes a NAS Mobility Registration Update). Four conditions trigger the fallback:

1. No active IKE SA exists when the handover notification is received.
2. MOBIKE was not negotiated during IKE_AUTH (the `MobikeSupported` flag is false).
3. The MOBIKE request times out (5 s timer with no response from the target N3IWF).
4. The target N3IWF replies with `INVALID_IKE_SPI`, indicating that the IPsec state synchronization failed and the target does not recognize the SA.

In all cases, the UE initiates `IKE_SA_INIT` + `IKE_AUTH` from scratch with the target N3IWF. The fallback is transparent to the Near-RT RIC and xApp: the source N3IWF still receives the *UE Context Release Command* from the AMF once the UE completes registration on the target, and the RC handover result is delivered normally.

5.3.6 Downlink Packet Buffering

The downlink packet buffer is implemented in the `internal/hobuffer/` package. It uses a per-TEID ring buffer to hold GTP-U packets during the handover transition.

```
type Manager struct {
    mu      sync.Mutex
    buffers map[uint32]*buffer // keyed by GTP-U TEID
    capacity int
}

func NewManager(capacity int) *Manager
func (m *Manager) Activate(teid uint32)
func (m *Manager) Enqueue(teid uint32, pkt []byte) bool
func (m *Manager) Flush(teid uint32) [][]byte
func (m *Manager) FlushAndWrite(teid uint32,
    writeFn func(pkt []byte) error) int
func (m *Manager) Cancel(teid uint32)
func (m *Manager) CleanupExpired(ttl time.Duration)
```

Listing 5.2: Downlink buffer Manager API

The buffer lifecycle during handover on the *target* N3IWF is as follows:

1. **Activate**: Called when the target N3IWF processes the *Handover Request* from the AMF and successfully sets up GTP-U tunnels for the admitted PDU Sessions. Creates an empty ring buffer (default capacity: 65536 packets) for each admitted GTP-U TEID.

2. **Enqueue**: Called by the NWu downlink forwarding goroutine when a GTP-U packet arrives from the UPF for a buffered TEID (see Section 4.5.2 for the rationale). If the buffer is full, the oldest packet is silently dropped.

3. **FlushAndWrite**: Called after the UE completes the MOBIKE `UPDATE_SA_ADDRESSES` exchange (Section 2.3.2), which reinstalls the XFRM rules with the correct peer outer IP. All buffered packets are then written to the IPsec tunnel in FIFO order.

4. **CleanupExpired**: A periodic cleanup routine removes orphaned buffers (from abandoned handovers) older than a configurable TTL (default: 60 seconds).

5.3.7 UE Context Release on Source N3IWF

After the target N3IWF sends the *Handover Notify* to the AMF, the AMF sends a *UE Context Release Command* to the source N3IWF to clean up the old UE context.

The source N3IWF's `HandleUEContextReleaseCommand` marks the handover as completed, releases the IKE SA and RAN UE context (deleting all XFRM states and policies), and sends the *UE Context Release Complete* to the AMF.

DPD suppression during handover. During the execution phase, the UE is switching Wi-Fi associations and is temporarily unreachable from the source N3IWF. IKEv2 Dead Peer Detection (DPD) timeouts can fire during this window, which would normally trigger a `UEContextReleaseRequest` to the AMF. However, if the source N3IWF detects that a handover is in progress, it suppresses this message. Without suppression, the AMF could receive the release request and the *Handover Notify* (from the target N3IWF) nearly simultaneously, leading to conflicting PFCP modifications on the same PDU Sessions at the UPF. The UE context on the source is instead cleaned up by the *UE Context Release Command* that the AMF sends after processing the *Handover Notify*.

5.3.8 Network Offload Workarounds

Linux network segmentation offloads on XFRM interfaces caused silent data-plane failures in the containerized testbed and required a targeted workaround.

TCP Segmentation Offload (TSO), Generic Receive Offload (GRO), and Generic Segmentation Offload (GSO) on `xfrmi*` and `ipsec0` interfaces cause the kernel to coalesce or segment packets *before* they enter the ESP encapsulation path. The resulting ESP packets exceed the tunnel MTU after encryption overhead is added, leading to fragmentation or silent drops inside the GTP-U tunnel.

The N3IWF container uses a custom entrypoint script that starts the N3IWF process in the background, waits for the `xfrmi*` interfaces to appear (they are created at runtime when the first UE connects), and then disables all three offloads:

```
1 ethtool -K "$iface" gro off gso off tso off
```

Listing 5.3: XFRM offload disabling

The behavior is controlled by the `N3IWF_DISABLE_OFFLOADS` environment variable (default: enabled).

5.4 AMF Modifications

The free5GC AMF (Section 2.2.3) does not natively support N2 handover for non-3GPP access. The following subsections describe the modifications introduced in the `amfCustom` fork, mapped to the handover flow shown in Figures 4.4 and 4.5 and to the design choices discussed in Section 4.5.

5.4.1 N3IWF Target ID Support

The base AMF only handles gNB targets in the *Handover Required* path: it assumes the `GlobalRANNodeID` inside `TargetRANNodeID` is always a `GlobalGNBID` and dereferences the corresponding pointer unconditionally.

The fix adds a `switch` on `targetID.Present` and on `GlobalRANNodeID.Present` in `handleHandoverRequiredMain` to also accept `GlobalN3IWFID`. The N3IWF encodes its Target ID as `TargetIDPresentTargetRANNodeID` with a `GlobalN3IWFID` inside (and an explicit `SelectedTAI`), reusing the standard NGAP structure rather than the `ChoiceExtensions` alternative.

Additionally, the AMF extracts the target N3IWF's IPv4 address from the SCTP remote address of the target RAN connection (`extractIPv4FromRanAddr`) and forwards it to the SMF as a hint for UPF N3 endpoint selection (see Section 5.5.3).

5.4.2 AMF UE NGAP ID Patching

The AMF allocates a new `AmfUeNgapId` for the target RAN UE context. Since the Source-to-Target container was built by the *source* N3IWF using its own AMF UE NGAP ID, the AMF must patch it before relaying. In `SendHandoverRequest`, if the inner `RRCContainer` starts with `{` (JSON), it decodes the object, overwrites the `amfUeNgapId` field with `targetUe.AmfUeNgapId`, re-encodes, and replaces the APER bytes. This ensures the target N3IWF's imported context uses the correct AMF UE NGAP ID.

5.4.3 Kn3iwf Security Key Refresh

During a Mobility Registration Update over non-3GPP access, the AMF must re-derive the Kn3iwf (the AN-level security key for non-3GPP access, defined in TS 33.501 [23]) aligned with the current uplink NAS COUNT. Without this refresh, the target N3IWF verifies IKE_AUTH with a stale Kn3iwf, resulting in `AUTHENTICATION_FAILED`.

The fix adds a call to `ue.DerivateAnKey(anType)` in `handleMobilityRegistrationUpdate` when the access type is `NON_3_GPP_ACCESS` and the security context is valid. This is performed *before* processing the registration request.

5.4.4 Stale Handover State Recovery

Several defensive fixes address scenarios where a previous handover failed mid-procedure (e.g., due to a PFCP timeout at the UPF):

Stale TargetUe link. In the base free5GC AMF, if a previous handover failed mid-procedure (e.g., PFCP timeout), the `sourceUe.TargetUe` reference is never cleared because no failure path invokes `DetachSourceUeTargetUe`. On the next handover attempt, the AMF detects the existing link and rejects the request with "Handover Required Duplicated", permanently blocking further handovers for that UE. The fix clears the stale reference before processing the new Handover Required.

Handover Notify failure rollback. In the base free5GC AMF, if `SendUpdateSmContextN2HandoverComplete` fails for any PDU Session during the Handover Notify processing (e.g., PFCP timeout), the error is only logged and the handover completes with a broken data plane. The fix tracks failures across all sessions and, if any occur, rolls back: it sends `N2HandoverCanceled` to the SMF for all sessions, detaches the source–target UE link, releases the target UE context via *UE Context Release Command*, and emits a failure metric. The source UE remains attached, allowing a subsequent handover attempt.

5.4.5 Target AN IP Hint to SMF

The `SendUpdateSmContextN2HandoverPreparing` function gains a `targetAnIpHint` parameter. Since the free5GC

`Nsmf_PDUSession_UpdateSMContext` OpenAPI schema does not expose a field for the target AN address, and modifying the auto-generated client and server code would have been significantly more invasive, the hint is passed in-band: a small header consisting of a magic marker (`SMFHO1`), a one-byte IP length, and the target AN IPv4 address is prepended to the existing N2SmInfo binary payload, which is already treated as an opaque byte array by the API. On the SMF side, if the payload starts with the magic marker, the target IP is extracted and the prefix is stripped before decoding the NGAP content; otherwise, the payload is processed normally. The extracted address is used to select the correct UPF N3 endpoint (see Section 5.5.3).

5.5 SMF Modifications

The free5GC SMF (Section 2.2.4) implements the standard path switch (UPF update at `HoState_COMPLETED`). The `smfCustom` fork introduces the early path switch and addresses several bugs encountered during N3IWF handover.

5.5.1 Early Path Switch at HoState_PREPARED

In the base SMF, the downlink FAR is updated to point to the target AN tunnel at `HoState_COMPLETED` (after Handover Notify). With N3IWF handover, once the UE performs MOBIKE, the source's XFRM outbound rules are broken; any downlink packets still arriving at the source are undeliverable.

The fix moves the DL FAR update to `HoState_PREPARED`: after `HandleHandoverRequestAcknowledgeTransfer` updates the `ANInformation` with the target's GTP-U endpoint, the SMF immediately sets `DLPDR.FAR.ApplyAction = {Forw: true}` and sends a PFCP Session Modification to the UPF. At `HoState_COMPLETED`, no DL FAR path-switch PFCP modification is needed; the session state is finalized.

5.5.2 DL Buffering Removal

The base SMF sets the DL FAR to `Buff=true, Nocp=true` when `UpCnxState == DEACTIVATED` during `HoState_PREPARING`. In the testbed, this triggered a Downlink Data Report (DLDR) / N1N2Transfer storm that overwhelmed the UPF, causing the SMF to interpret the delayed PFCP responses as UPF unavailability (see Section 4.5.2 for details). The fix removes the PFCP Modification for DL buffering entirely during PREPAR-

ING. The UPF continues forwarding to the old tunnel until the path switch at HoState_PREPARED redirects traffic to the target.

5.5.3 Multi-Homed UPF N3 Endpoint Selection

The base SMF always selects the first configured N3 interface (N3Interfaces[0].IP()) for the GTP-U tunnel endpoint. In a multi-homed UPF with N3 IPs on different subnets, this may select an address unreachable from the target N3IWF.

A new method SelectIPForAN on UPFInterfaceInfo picks the UPF N3 endpoint whose IPv4 address shares the same /24 prefix as the target AN IP. Both BuildPDUSessionResourceSetupRequestTransfer and BuildPathSwitchRequestAcknowledgeTransfer use ctx.TargetANIP (extracted from the AMF hint) to call SelectIPForAN, falling back to the first endpoint if no match is found. This was necessary because in the testbed the first N3IWF shares a Docker bridge network with the UPF (co-located on the same host), while the second N3IWF is on a separate host reachable only through a different subnet.

5.5.4 Target AN IP Hint Unwrapping

HandleHandoverRequiredTransfer unwraps the SMFHO1 magic prefix injected by the AMF (Section 5.4.5), extracts the target AN IPv4 address, and stores it in ctx.TargetANIP. It also stores the TargetAccessType and TargetRanNodeID from the update data, enabling the SMF to detect access-type changes (e.g., 3GPP to non-3GPP) and update ctx.AnType at HoState_COMPLETED.

The SMContext struct gains three new fields: TargetAccessType, TargetRanNodeID, and TargetANIP. All three are cleared at HoState_COMPLETED or HoState_CANCELLED.

5.5.5 Stale Handover State Recovery

If a previous handover failed mid-procedure (e.g., PFCP timeout), the SMContext can be left in ModificationPending with a non-empty HoState. Any subsequent handover attempt would deadlock waiting for the state to become Active.

The fix adds recovery logic at the entry of HoState_PREPARING, HoState_PREPARED, and HoState_CANCELLED: if the current state is ModificationPending, it is forcibly reset to Active. At HoState_PREPARING, any stale handover fields (HoState,

`TargetAccessType`, `TargetRanNodeID`, `TargetANIP`, forwarding tunnels) are cleared. On `SessionUpdateFailed`, all handover state is reset so the next attempt can proceed cleanly.

Chapter 6

Experimental Results and Analysis

This chapter describes the experimental testbed, the methodology, and the test scenarios used to evaluate the proposed closed-loop handover mechanism for non-3GPP access. Section 6.1 presents the physical testbed, its network configuration, and the test methodology. Section 6.2 validates functional correctness, characterizes handover timing, and verifies the accuracy of the KPM throughput metrics against application-layer measurements. Section 6.3 quantifies the data-plane impact during handover. Section 6.4 provides a comparative analysis.

6.1 Experimental Setup

6.1.1 Testbed Topology

The testbed comprises four nodes deployed in a university laboratory environment, interconnected via Ethernet and Wi-Fi. Three nodes are bare-metal machines with two network interfaces each; the fourth is a virtual machine hosted on a Proxmox cluster within the same campus network with a virtual network interface connected to the campus network.

A distinguishing characteristic of this testbed is that the Wi-Fi access points are *not* standalone devices: each N3IWF host (Dell Edge Gateway 5200 and UP Xtreme i12 Edge 1) also operates a Wi-Fi NIC in AP mode via `hostapd`. The UE node (UP Xtreme i12 Edge 2) uses an identical Wi-Fi NIC in station (STA) mode. All three Wi-Fi interfaces use the Intel AX210 (Wi-Fi 6E) chipset.

Although the Intel AX210 chipset supports Wi-Fi 6E, the testbed op-

Table 6.1: Testbed hardware specifications

Node	**Role**	**CPU**	**RAM**
Dell Edge Gateway 5200	5GC, N3IWF-A, E2Node, AP-A	Intel i5-9500TE (6C)	16 GB
UP Xtreme i12 Edge 1	N3IWF-B, E2Node, AP-B	Intel i7-1270PE (16C)	16 GB
UP Xtreme i12 Edge 2	N3IWUE (UE)	Intel i7-1270PE (16C)	16 GB
nrrtric	Near-RT RIC, xApp	QEMU vCPU (8C)	8 GB

erates on the 2.4 GHz band using 802.11n. This choice is aligned with the N3IWF non-3GPP setting, where access technologies are heterogeneous, and 802.11n offers broad interoperability across device classes and generations. Moreover, the goal of this testbed is to evaluate the 5G control-plane behavior and the O-RAN closed-loop workflow, rather than to benchmark PHY/MAC throughput. Under this scope, the selected Wi-Fi generation does not materially affect the KPM-to-decision mapping or the handover-control conclusions reported in this chapter.

Figure 6.1 illustrates the testbed topology, showing the Ethernet management network and the two Wi-Fi segments.

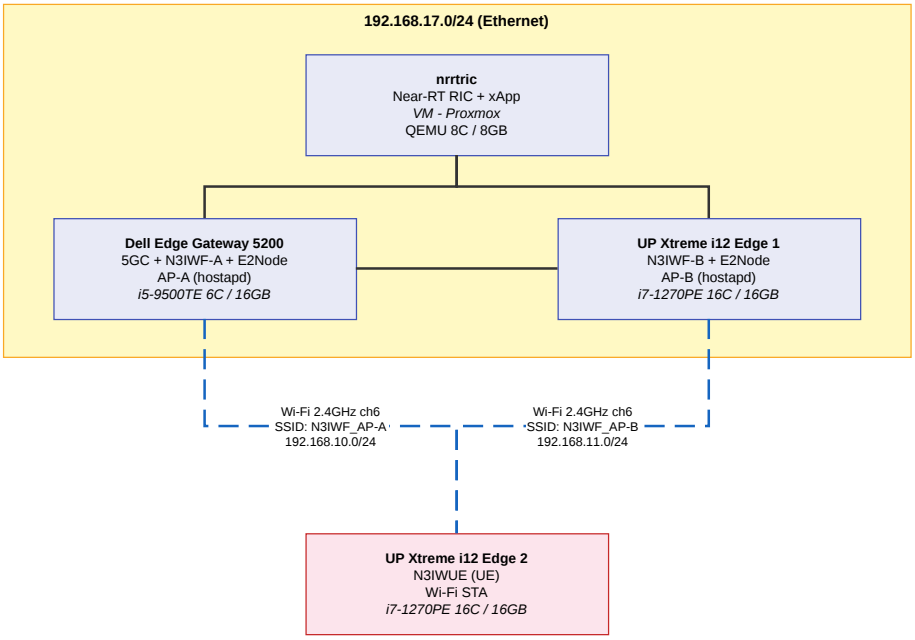

Figure 6.1: Testbed topology. Solid lines represent the Ethernet management/control network; dashed lines represent Wi-Fi segments. The UE associates with one AP at a time and switches during handover.

6.1.2 Component Deployment

All 5G Core network functions and N3IWF instances are deployed as Docker containers. Figure 6.2 shows the logical deployment: which software components run on each physical node and the interfaces connecting them. Table 6.2 summarizes the deployment across the four nodes.

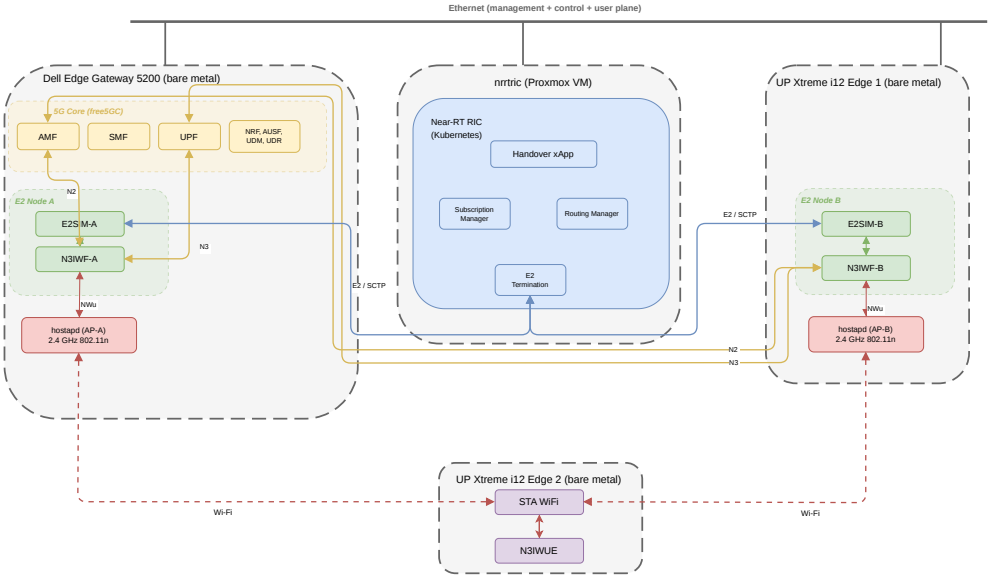

Figure 6.2: Logical component deployment across testbed nodes. Dashed boxes represent physical hosts. Docker containers (5GC NFs, N3IWFs, E2SIMs) and the Kubernetes-based Near-RT RIC are connected via Ethernet; Wi-Fi links (dotted) connect the APs to the UE.

The system is built on the free5GC open-source 5G Core with custom modifications to the AMF, SMF, and N3IWF to support the N2-based handover procedure described in Chapter 4. The Near-RT RIC uses the O-RAN Software Community (OSC) platform and hosts the Handover xApp as a Kubernetes pod.

The UPF container on Dell Edge Gateway 5200 additionally hosts an `iperf3` server bound to the data-network interface (`10.100.200.17`), used as the traffic endpoint for most of the throughput measurements. The N3IWUE runs as a native process on UP Xtreme i12 Edge 2 to allow direct management of the Wi-Fi interface and `NetworkManager`. Each N3IWF instance runs a co-located E2SIM process that connects to the Near-RT RIC over the Ethernet network.

6.1.3 Technologies and Platforms

Table 6.3 summarizes the key technologies and platforms involved in this work and their role in the system.

Table 6.2: Component deployment across testbed nodes

Node	**Deployment**	**Components**
Dell Edge Gateway 5200	Docker Compose	AMF, SMF, UPF, NRF, AUSF, UDM, UDR, PCF, NSSF, NEF, CHF, N3IWF-A, E2Node-A, Wi-Fi metrics exporter
UP Xtreme i12 Edge 1	Docker Compose	N3IWF-B, E2Node-B, Wi-Fi metrics exporter
UP Xtreme i12 Edge 2	Native process	N3IWUE
nrrtric	Kubernetes	Near-RT RIC platform (E2 Termination, Subscription Manager, Routing Manager), Handover xApp

Table 6.3: Technologies and platforms used in this thesis

Technology	**Role in this thesis**
free5GC [16]	Open-source 5G Core (Go): AMF, SMF, UPF, N3IWF, and supporting NFs. Custom modifications to AMF, SMF, and N3IWF.
O-RAN SC Near-RT RIC	O-RAN Software Community platform (j-release) hosting the E2 Termination, Subscription Manager, and xApps on Kubernetes.
E2SIM [50]	O-RAN SC E2 simulator, extended with regenerated ASN.1 for E2AP v3.01, E2SM-KPM v3, E2SM-RC v1.03.
xDevSM [31]	Framework for xApp development against E2SM service models (Python).
IKEv2 / IPsec / MOBIKE	Security tunneling (RFC 7296, RFC 4301, RFC 4555) between UE and N3IWF over untrusted Wi-Fi.
IEEE 802.11 (Wi-Fi)	Non-3GPP access technology (2.4 GHz, 802.11n) used by the UE to reach the N3IWF.
Docker / Kubernetes	Containerization of 5GC NFs and N3IWF instances (Docker Compose); Near-RT RIC platform deployment (Kubernetes).

6.1.4 Network Configuration

The testbed uses four IP subnets, summarized in Table 6.4.

Table 6.4: Network subnets

Subnet	Medium	Purpose
192.168.17.0/24	Ethernet	Cross-host: N2/N3 for N3IWF-B, E2AP, management
192.168.10.0/24	Wi-Fi (AP-A, Dell Edge Gateway 5200)	IKEv2/IPsec between UE and N3IWF-A
192.168.11.0/24	Wi-Fi (AP-B, UP Xtreme i12 Edge 1)	IKEv2/IPsec between UE and N3IWF-B
10.100.200.0/24	Docker bridge (`br-free5gc`)	Core NFs

On Dell Edge Gateway 5200, all core NFs and N3IWF-A communicate over the Docker bridge network `br-free5gc` (`10.100.200.0/24`), which carries SBI, NGAP, PFCP, and GTP-U traffic locally. N3IWF-B on UP Xtreme i12 Edge 1 reaches the AMF and UPF over the Ethernet subnet (`192.168.17.0/24`), which also carries E2AP traffic between the E2Nodes and the Near-RT RIC. All communication between the UE and the serving N3IWF traverses the Wi-Fi link. The N3IWF then forwards the decrypted user-plane packets in GTP-U towards the UPF.

Wi-Fi access point configuration Both access points are configured via `hostapd` with the parameters listed in Table 6.5.

Table 6.5: `hostapd` configuration for the two access points

Parameter	Access Point
SSID	`N3IWF_AP-A` / `N3IWF_AP-B`
Band / Mode	2.4 GHz, 802.11n (`hw_mode=g`)
Channel	6
Security	WPA2-PSK / CCMP
HT Capabilities	SHORT-GI-20, SHORT-GI-40, HT40+
WMM	Enabled

The UE uses a static IP address on each Wi-Fi subnet, avoiding DHCP-related delays during handover.

6.1.5 Test Scenarios and Methodology

The evaluation is structured around three test scenarios, each designed to isolate or combine different aspects of the proposed system. All scenarios use `iperf3` to generate controlled traffic between the UE and the UPF-hosted server.

Traffic parameters Each test scenario is repeated across two target bitrates (60 and 100 Mbps) using UDP, in both downlink (DL, server-to-UE) and uplink (UL, UE-to-server) directions. Each combination is repeated for three independent runs of 30 seconds each.

KPM subscription parameters Throughout all experiments, the Handover xApp subscribes to both E2Nodes with a `reportingPeriod` of 1 s and a `granularityPeriod` of 1 s. Each KPM Indication therefore carries one second of accumulated per-UE measurements, and the xApp receives one report per node per second.

Metrics The following metrics are collected for each run:

- **Throughput** (Mbps): average and per-second throughput as reported by `iperf3`.
- **Packet loss** (%): for UDP, the fraction of lost datagrams over total sent.
- **Jitter** (ms): the inter-packet arrival variation.

In addition, for handover scenarios, the following timing metrics are extracted from N3IWF and N3IWUE logs (timestamped with NTP-synchronized clocks):

- **Handover latency** (ms): end-to-end time from xApp control request to Handover Notify.
- **Wi-Fi switch time** (ms): time between the UE initiating the Wi-Fi switch and completing re-association on the target AP.
- **Service interruption** (s): duration of zero throughput during handover, as observed in `iperf3` per-second reports.

Test scenarios

1. **Baseline (no handover)**: The UE is connected to a single N3IWF (either N3IWF-A or N3IWF-B) for the entire duration of the test. No handover is triggered. This establishes the reference throughput and packet loss for each path independently, allowing comparison with the handover scenarios.

2. **With handover (full system)**: The complete closed-loop system is active. The xApp monitors KPM reports from both E2Nodes and triggers an N2-based handover via RC Control Request when it detects a load imbalance. The N3IWF performs downlink buffering (HOBuffer), and the N3IWUE uses the on-demand Wi-Fi rescan optimization. This scenario measures the end-to-end handover impact on ongoing traffic.

3. **Without state sync**: Same as scenario 2, but both the IPsec state transfer and the downlink buffer are disabled. Without the IKE/Child SA context, MOBIKE fails and the UE falls back to a full IKE re-establishment; without the DL buffer, packets arriving from the UPF during the transition window are dropped. This scenario represents the degraded fallback and isolates the combined contribution of state sync and buffering to handover performance.

6.2 Functional Correctness and Handover Characterization

This section demonstrates that the entire closed-loop system operates correctly end-to-end: the xApp receives KPM telemetry, decides to trigger a handover, the N2-based handover executes successfully, and data-plane traffic resumes on the target N3IWF. We also present a detailed timing breakdown of the handover phases and characterize the Near-RT RIC control-loop latency.

Throughout this section, two events serve as temporal boundaries for the handover procedure. The **HO Trigger** denotes the RIC Control Request issued by the xApp to initiate the handover. The **Handover Notify** denotes the N2 message sent by the target N3IWF to the AMF upon successful UE re-attachment, marking the completion of the procedure (Section 2.2.6). All handover latency measurements in this section are bounded by these two events.

6.2.1 KPM Throughput Shift

To qualitatively verify that the KPM telemetry observed by the xApp correctly follows the UE migration across N3IWFs, Figure 6.3 reports the `DRB.UEThpDl` metric received from the source and target E2Nodes, aligned to the handover trigger. The curves are aggregated over a dedicated telemetry validation session of 20 handover events conducted under a constant `iperf3` UDP downlink load of 50 Mbps, and plotted relative to the RIC Control Request ($t = 0$). This session is distinct from the main evaluation campaign of 127 handovers described in Sections 6.3 and 6.4, which follows the traffic parameters defined in Section 6.1.5.

Before the trigger, only the source E2Node reports non-zero downlink throughput, while the target remains at zero, as no UE is associated to it. Immediately after the trigger, the source-side throughput collapses as the UE leaves the source N3IWF. The target-side throughput becomes visible only after the handover completes and the first post-handover KPM indications are generated by the target E2Node, producing the rightward shift visible in the figure. A brief peak is also visible when the target-side throughput first appears; this is consistent with the post-handover transient caused by the flush of downlink packets buffered during the transition window.

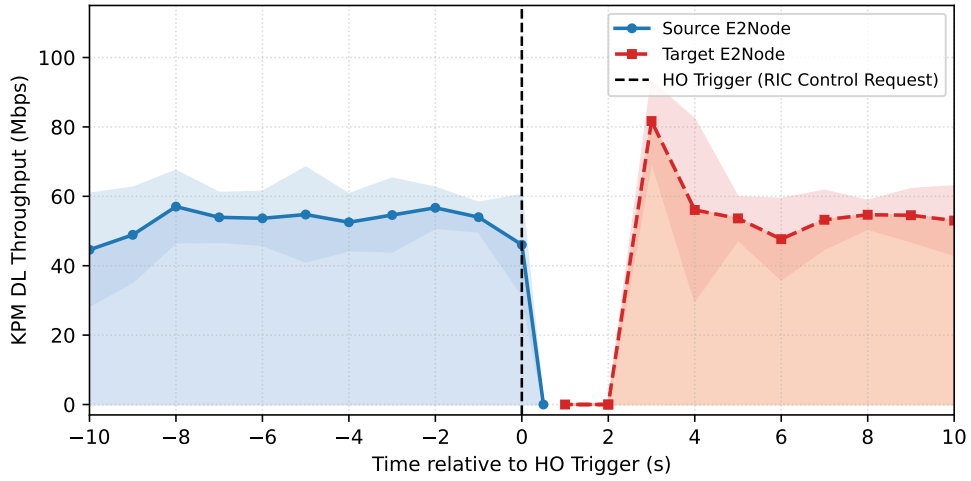

Figure 6.3: Aggregated KPM downlink throughput shift during handover, aligned to the HO trigger ($t = 0$). Source-side `DRB.UEThpDl` collapses after the RIC Control Request; target-side throughput appears only after handover completion and the first subsequent KPM Indication from the target E2Node. Shaded bands indicate ± 1 standard deviation across the events contributing to both source and target series.

The KPM shift latency is further quantified over the full evaluation campaign of 127 handovers. It is defined as the time from the RIC Control Request to the first target-side KPM Indication reflecting the migrated UE in the Format 3 report list, and measures 2.70 ± 0.47 s (median: 2.64 s, min: 1.9 s, max: 4.9 s). This latency decomposes into three contributions: the handover execution itself ($\sim$815 ms, see Section 6.2.4), the 500 ms `rc_metrics.json` refresh interval on the target E2Node, and up to one full 1 s KPM reporting period before the next Indication is sent. The observed values are consistent with these contributing factors.

6.2.2 KPM Throughput Accuracy

Beyond verifying that the KPM pipeline correctly tracks UE presence, we validate the accuracy of the throughput metrics (`DRB.UEThpDl` and `DRB.UEThpUl`) by comparing them against simultaneous `iperf3` measurements during steady-state periods (i.e., excluding the handover transition window).

Because the E2SIM computes throughput from Wi-Fi station counters (`tx_bytes` and `rx_bytes` reported by `iw station dump`), the KPM values reflect MAC-layer throughput rather than application-layer goodput. The MAC counters include 802.11 frame headers, CCMP encryption overhead, A-MPDU sub-frame delimiters, and retransmitted frames, all of which are invisible to `iperf3`, which measures only the payload bytes delivered to the transport socket.

Table 6.6 summarises the results of a dedicated validation experiment: four 400-second UDP `iperf3` sessions at 100 Mbps are run in each direction (DL and UL) while the E2SIM reports KPM indications every second. The KPM throughput values are extracted from the `e2node.log` and aligned with the corresponding `iperf3` per-second intervals using absolute timestamps.

Table 6.6: KPM vs. `iperf3` throughput comparison (UDP, 100 Mbps, 400 s)

Metric	**Downlink**	**Uplink**
`iperf3` mean throughput (Mbps)	86.7 ± 9.0	100.0 ± 1.0
KPM mean throughput (Mbps)	97.3 ± 16.9	112.5 ± 11.4
Mean overhead (%)	13.4	12.5
Std dev overhead (%)	23.2	11.4
Volume ratio (KPM / `iperf3`)	1.122	1.125

The KPM throughput consistently exceeds the `iperf3` goodput by approximately 12–13%, with a cumulative volume ratio of $\approx 1.12\times$ in both directions. This overhead is attributable to the IPsec ESP encapsulation

(header, IV, padding, ICV), UDP/IP tunnel headers, and 802.11 MAC-layer framing (CCMP, A-MPDU delimiters), all of which are counted by the Wi-Fi station counters but are stripped before the payload reaches the `iperf3` socket. The overhead is symmetric across DL and UL and stable over the full measurement window, confirming that the Wi-Fi-to-KPM mapping described in Section 4.4.2 provides a consistent and predictable proxy for access-layer throughput, suitable for threshold-based decisions in xApps.

6.2.3 Throughput Continuity

Figure 6.4 shows the downlink GTP-U throughput on both N3IWFs across two consecutive handover events. The data is collected via `tcpdump` on each N3IWF container during a downlink `iperf3` session: traffic originates from the `iperf3` server, traverses the UPF, and is delivered to the serving N3IWF over GTP-U. The HO trigger and Handover Notify timestamps are extracted from the xApp and N3IWF logs respectively.

During HO #1, traffic flows through N3IWF-A at ∼105 Mbps. Upon the xApp trigger (dashed line), the throughput on N3IWF-A drops sharply and appears on N3IWF-B almost immediately: the early path switch redirects GTP-U traffic to the target during the preparation phase, well before the Handover Notify. During the window between this path switch and the UE completing the Wi-Fi switch and MOBIKE exchange on the target, the arriving packets are collected by the target's DL buffer rather than dropped. HO #2 reverses the direction with an identical pattern. In both cases, throughput fully recovers to the pre-handover level on the target, confirming that the GTP-U tunnel is correctly re-established and the buffered packets are delivered.

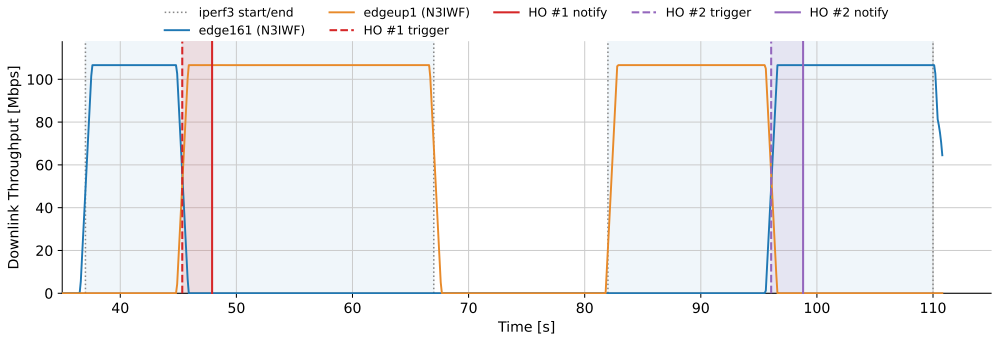

Figure 6.4: Downlink GTP-U throughput on both N3IWFs across two consecutive handovers.

6.2.4 Handover Timing Breakdown

To understand where latency is spent during the handover, we instrument both the N3IWF and the N3IWUE with phase-level timestamps. The phases correspond to the preparation (Figure 4.4) and execution (Figure 4.5) sequence diagrams presented in Chapter 4. Figure 6.5 presents the mean duration of each phase across 127 successful handovers with MOBIKE enabled (the full-system scenario). The phases are sequential and represent the end-to-end handover path from the xApp's RIC Control Request to the Handover Notify, following the execution flow illustrated in Figures 4.4 and 4.5. The first phase label, **T2S**, abbreviates the Target-to-Source Transparent Container (Section 5.3.4), the NGAP message that delivers the handover command from the network to the UE.

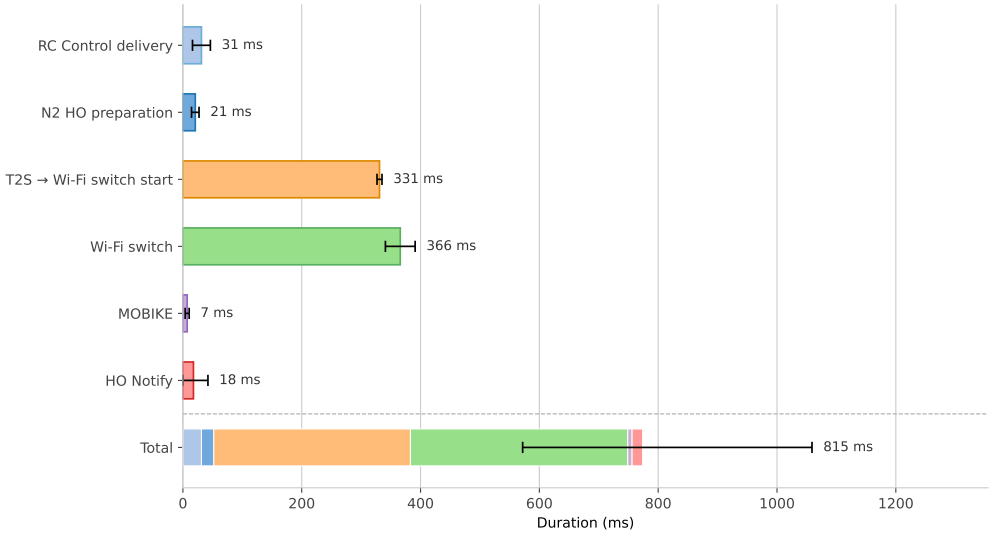

Figure 6.5: Handover timing breakdown with MOBIKE (state sync enabled). Each bar shows the mean duration; error bars indicate ±1 standard deviation. The two dominant phases are the T2S-to-Wi-Fi-switch delay (∼331 ms) and the Wi-Fi switch itself (∼366 ms).

The two dominant phases account for over 85% of the total handover time:

- **T2S → Wi-Fi switch start** (∼331 ms): This is the intentional delay introduced by the N3IWUE after receiving the T2S container (Section 5.3.4). The UE initiates an on-demand Wi-Fi scan (`nmcli device wifi rescan`) and waits 300 ms for the scan results to populate the NetworkManager AP cache. Without this optimization, a full directed

probe scan during `nmcli con up` adds 3–4 s to the handover. The low standard deviation (4.2 ms) confirms this phase is deterministic.

- **Wi-Fi switch** (∼366 ms): The actual Wi-Fi re-association from the source AP to the target AP, including WPA2 authentication. This phase exhibits moderate variability (range: 303–426 ms), reflecting the inherent non-determinism of 802.11 association procedures.

The remaining phases are comparatively fast:

- **RC Control delivery** (∼31 ms): The end-to-end latency from the xApp sending the RIC Control Request via RMR to the E2SIM forwarding it over HTTP to the N3IWF. This includes RMR message routing through the Near-RT RIC platform and the HTTP POST to the N3IWF's handover endpoint.
- **N2 HO preparation** (∼21 ms): The NGAP signaling round-trip between the source N3IWF, the AMF, and the target N3IWF (Handover Required → Handover Request → Handover Request Acknowledge → Handover Command). The low latency reflects the Ethernet-connected control plane.
- **MOBIKE** (∼7 ms): The `UPDATE_SA_ADDRESSES` exchange between the UE and the target N3IWF. Since the target N3IWF already possesses the full IKE SA and Child SA state (transferred in the Source-to-Target Container), this exchange consists of a single INFORMATIONAL request/response pair to update the IP addresses. The low latency (mean: 7.2 ms, max: 26.8 ms) confirms that MOBIKE is effectively a lightweight address update rather than a cryptographic renegotiation.
- **Buffer flush and HO Notify** (∼18 ms): The time to deliver the downlink packets that were buffered by the target N3IWF during the transition. The high variability (std: 24.4 ms, max: 119.7 ms) depends on the volume of buffered data, which in turn depends on the downlink bitrate and the duration of the buffering window. The end of this Buffer flush triggers the Handover Notify from target N3IWF to AMF.

The mean total handover time is 815 ms, with 95% of handovers completing within 929 ms. The long tail (max: 2439 ms) is attributable mainly to Wi-Fi association delays.

6.2.5 Control Loop Latency

The RC Control round-trip time (from sending the RIC Control Request to receiving the Acknowledge) across 127 handovers averages 31.2 ± 15.0 ms (P95: 71.0 ms), well within the 1 s Near-RT RIC budget. The xApp reaction time is negligible, as the handover decision is made synchronously within the KPM Indication callback.

The KPM subscription setup is also fast: the Subscription Manager REST API responds in 2.6–5.5 ms (mean: $\sim$4.0 ms), allowing the xApp to begin receiving indications within one reporting period after startup.

6.3 Data Plane Impact

This section quantifies the impact of the handover on the UDP downlink data plane and evaluates the effectiveness of the downlink buffering mechanism. We compare the full-system scenario (scenario 2, with state sync and DL buffer) against the without-state-sync scenario (scenario 3, where both IPsec state transfer and the DL buffer are disabled) at two target bitrates (100 Mbps and 60 Mbps) to isolate the effect of channel saturation on the buffer's ability to prevent packet loss.

All graphs are generated from raw `iperf3` UDP DL logs, aligned to the exact HO trigger timestamp (extracted from the xApp log), and averaged across all runs. The per-second throughput and packet loss are plotted on dual Y-axes, with ± 1 standard deviation shaded bands.

6.3.1 UDP DL at 100 Mbps

Figure 6.6 shows the UDP DL throughput and packet loss during handover with the DL buffer enabled at a target bitrate of 100 Mbps. At $t = 0$ (the HO trigger), the throughput begins to drop as the source N3IWF activates the buffer and the UE starts the Wi-Fi switch. The throughput reaches its minimum of approximately 43 Mbps in the second containing the trigger, while the peak packet loss reaches $\sim$17%. Throughput recovers to the nominal rate within approximately 2 s after the trigger.

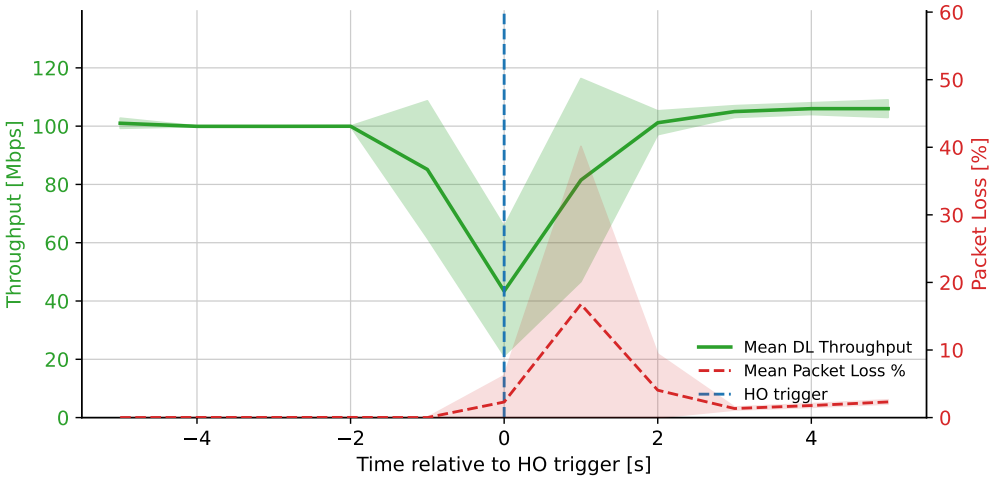

Figure 6.6: UDP DL impact during handover at 100 Mbps with DL buffer enabled.

Figure 6.7 shows the same measurement in the without-state-sync scenario (no IPsec state transfer, no DL buffer). The packet loss is dramatically higher: the peak reaches ∼56%, meaning more than half of the packets sent during the critical second are lost. Without state sync, MOBIKE cannot complete, so the target XFRM rules are never installed during the transition; packets arriving from the UPF during this window are silently dropped.

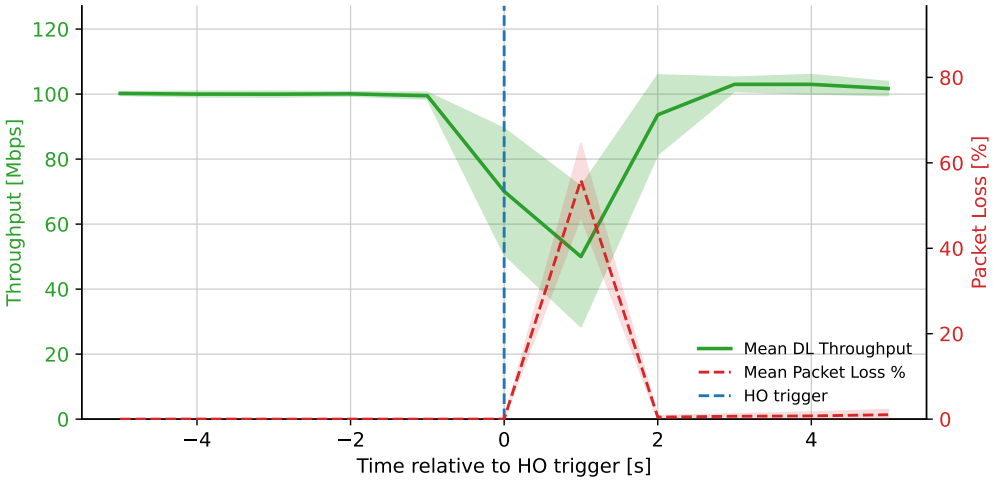

Figure 6.7: UDP DL impact during handover at 100 Mbps without state sync (no DL buffer).

The residual 17% packet loss with the buffer enabled at 100 Mbps is attributable to the interaction between the buffer flush and the Wi-Fi channel

capacity. When the buffered packets are flushed after the MOBIKE update, they compete with live traffic for the Wi-Fi link, which is already operating near saturation at 100 Mbps. Some packets are dropped at the Wi-Fi layer due to queue overflow.

6.3.2 UDP DL at 60 Mbps

To confirm that the residual loss at 100 Mbps is due to channel saturation rather than a buffer defect, we repeat the measurement at 60 Mbps, a bitrate well within the Wi-Fi channel capacity.

Figure 6.8 shows the result with the DL buffer enabled. The throughput dips briefly to ∼19 Mbps during the Wi-Fi switch (reflecting the physical disconnection), but the packet loss is negligible: the peak is only 0.39%. After the handover, a throughput overshoot is visible (∼80–95 Mbps for 2–3 s) as the buffer flush delivers the accumulated packets alongside live traffic. The channel has sufficient bandwidth to absorb this burst without packet drops.

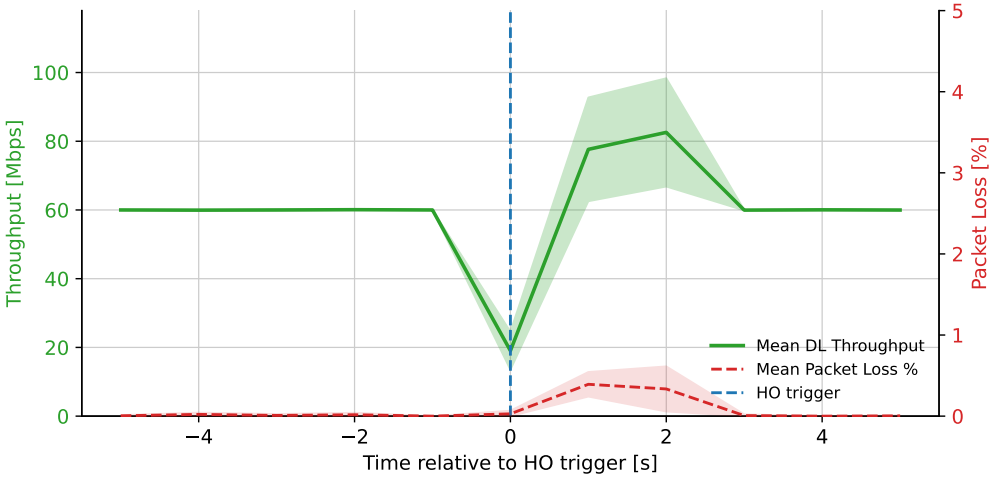

Figure 6.8: UDP DL impact during handover at 60 Mbps with DL buffer. Packet loss is negligible (<0.4%), confirming the buffer mechanism works correctly when the channel is not saturated.

Figure 6.9 shows the without-state-sync scenario at 60 Mbps. Despite the lower bitrate, the packet loss spikes to ∼54%, comparable to the 100 Mbps without-state-sync case. This confirms that the packet loss without state sync is determined by the handover duration (during which all DL packets are lost), not by the bitrate.

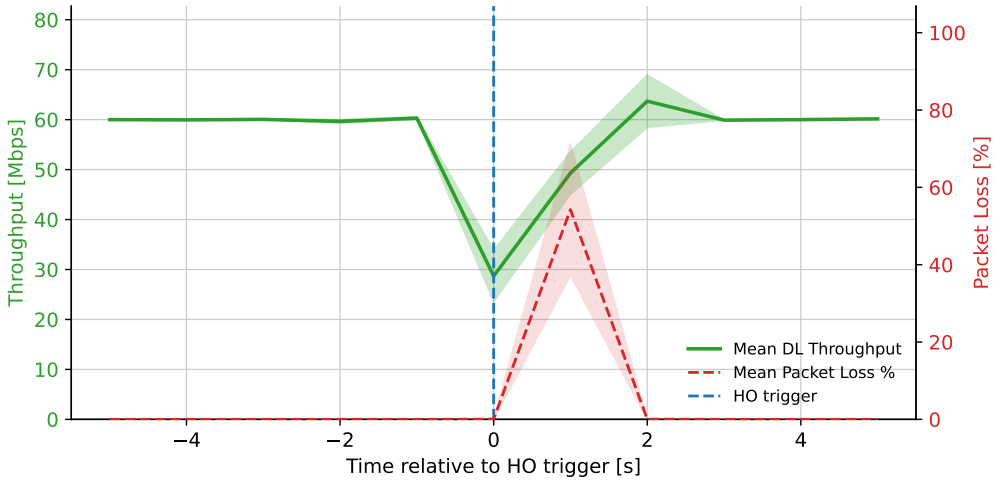

Figure 6.9: UDP DL impact during handover at 60 Mbps without state sync (no DL buffer). Packet loss peaks at ∼54%, similar to the 100 Mbps without-state-sync case.

6.3.3 Summary

Table 6.7 summarizes the key metrics across all four scenarios.

Table 6.7: Data plane impact summary. **FS** = Full system (with state sync and DL buffer); **NS** = Without state sync (no DL buffer). Avg and Max throughput are measured in steady state, excluding the handover disruption window.

Scenario	**Rate**	**Loss (%)**	**Min (Mbps)**	**Avg (Mbps)**	**Max (Mbps)**
FS	100 Mbps	16.8	43.4	102.1	106.0
NS	100 Mbps	56.0	50.0	100.8	103.0
FS	60 Mbps	0.4	19.0	60.0	82.6
NS	60 Mbps	54.3	28.7	60.0	63.7

The results lead to two conclusions. First, the DL buffering mechanism is effective: at 60 Mbps it virtually eliminates packet loss (0.4% vs. 54.3%), and at 100 Mbps it reduces the peak loss by a factor of 3.3× (16.8% vs. 56.0%). Second, the buffer's effectiveness is bounded by the Wi-Fi channel capacity: at high bitrates, the buffer flush competes with live traffic for a saturated link, resulting in residual loss. This is an inherent limitation of the physical layer, not a deficiency of the buffering mechanism itself.

In the uplink direction, the DL buffer has no effect: the UE cannot transmit while disconnected from Wi-Fi, so uplink service interruption is deter-

mined solely by the Wi-Fi switch and reconnection time, independently of the buffer setting.

6.4 Comparison

This section compares the full-system handover (with IPsec state sync and MOBIKE) against the fallback scenario (without state sync, requiring full IKE re-establishment), and establishes that the handover mechanism introduces no measurable overhead outside the disruption window. The without-state-sync scenario does not use the DL buffer, as the buffer relies on the state-sync mechanism to flush packets to the target.

6.4.1 With State Sync vs. Without State Sync

When the IPsec state transfer is disabled, the target N3IWF does not receive the IKE SA and Child SA context from the source. The UE still performs the Wi-Fi switch and attempts a MOBIKE `UPDATE_SA_ADDRESSES` exchange, but this fails because the target has no matching IKE SA. The UE then falls back to a full IKE re-establishment (IKE_SA_INIT + IKE_AUTH) followed by NAS re-registration to recover the PDU Session.

Figure 6.10 shows the handover timing breakdown for both scenarios side by side. In the without-state-sync case, the MOBIKE phase is replaced by two additional phases:

- **IKE re-establishment** (∼71 ms): a full IKE_SA_INIT and IKE_AUTH exchange with the target N3IWF, including Diffie-Hellman key exchange, EAP-5G authentication, and Child SA creation.
- **NAS re-registration** (∼261 ms): the UE re-registers with the 5G Core through the target N3IWF, including NAS Registration Request/Accept and PDU Session re-establishment.

The combined effect is a total handover time that approximately doubles: from ∼815 ms (with state sync) to ∼1 961 ms (without). More critically, the without-state-sync scenario proved unstable in extended testing: after several dozen consecutive handovers, the AMF began failing to process the repeated full re-registrations, resulting in handover failures. This prevented the collection of a full dataset comparable to the with-state-sync scenario and shows that the IPsec state transfer is not merely a performance optimization but a prerequisite for reliable operation under repeated handovers.

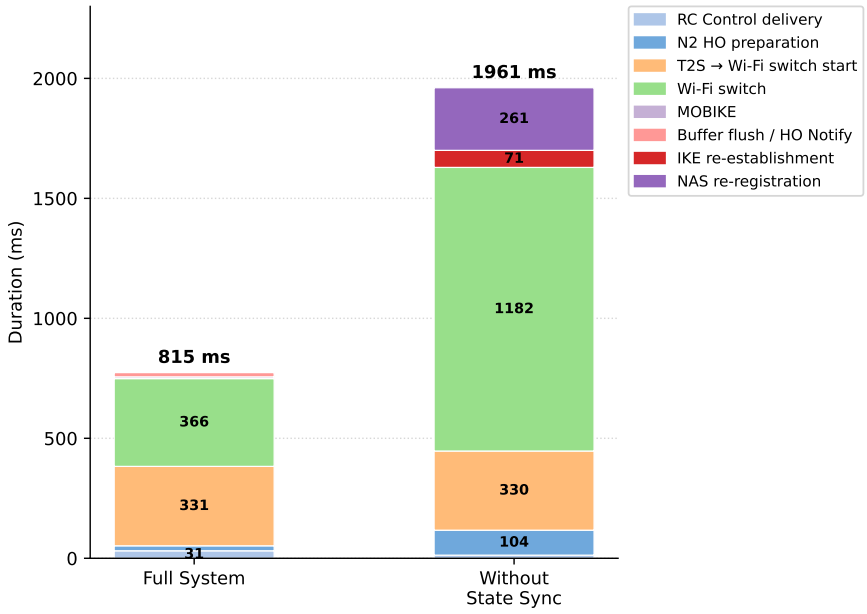

Figure 6.10: Handover timing breakdown: full system (with state sync) vs. without state sync. Each bar is a single stacked bar where each color corresponds to a handover phase, as indicated in the legend. The grey segment in the full-system bar represents inter-phase overhead not attributable to any single measured phase. Removing state sync replaces the 7 ms MOBIKE exchange with IKE re-establishment (∼71 ms) and NAS re-registration (∼261 ms), and extends the Wi-Fi switch from ∼366 ms to ∼1 182 ms, causing the total to approximately double.

6.4.2 Baseline Comparison

To verify that the handover mechanism does not introduce persistent overhead on the data plane, we compare the full-system scenario (with handovers occurring during the run) against the baseline scenario (no handovers, UE connected to a single N3IWF for the entire run). Table 6.8 presents the UDP DL results at 100 Mbps.

Table 6.8: Baseline vs. full system: UDP DL at 100 Mbps (30 s runs)

Scenario	**Throughput (Mbps)**	**Packet Loss (%)**	**Jitter (ms)**
Baseline (no HO)	100.0 ± 0.0	≈ 0.00	0.13 ± 0.03
Full system (with HOs)	98.4 ± 0.5	1.42 ± 0.53	0.15 ± 0.04

The baseline achieves exactly 100 Mbps with zero packet loss, confirming

that the testbed's Wi-Fi and GTP-U path can sustain the target rate. The full-system scenario shows a marginal throughput reduction (98.4 Mbps, i.e., 1.6% below baseline) and a 1.42% average packet loss over the 30 s run. Crucially, this loss is entirely concentrated in the $\sim$2 s window around each handover event, as demonstrated by the per-second analysis in Section 6.3. Outside the handover window, the per-second throughput is indistinguishable from the baseline.

The jitter values are comparable between the two scenarios (0.13 ms vs. 0.15 ms), confirming that the handover mechanism and the O-RAN control loop (KPM subscriptions, periodic metric collection) do not introduce measurable jitter in steady-state operation.

Chapter 7

Conclusions and Future Work

The objective of this thesis was to bring non-3GPP access nodes under the same O-RAN monitoring and control framework that already serves 3GPP RAN elements, by integrating the N3IWF as an E2 Node, and to validate this integration through a concrete control use case: RIC-driven N2-based handover between N3IWF instances.

Non-3GPP access nodes such as the N3IWF do not natively implement E2AP and are invisible to the Near-RT RIC. To address this, a co-located E2 agent was designed and implemented, exposing the N3IWF as an E2 Node with E2SM-KPM v3 for per-UE Wi-Fi monitoring and E2SM-RC v1.03 for RIC-issued control actions. The resulting telemetry pipeline bridges the gap between Wi-Fi radio statistics and the 3GPP measurement framework, so that xApps can consume metrics from both access technologies through a single, uniform interface.

To validate the control path, the thesis adapted the standard 3GPP N2 handover to the non-3GPP context. Unlike classical 3GPP mobility, this scenario lacks an RRC layer, does not provide an Xn interface between access nodes, and relies on IPsec outer IP addresses rather than radio bearers for downlink packet delivery. The proposed mechanism transfers the full IKE SA and Child SA state between N3IWF instances within the Source-to-Target Transparent Container, enabling the target to immediately resume secure user-plane processing. The execution phase leverages MOBIKE to update tunnel endpoints after the UE switches Wi-Fi access points, while a target-side downlink buffer mitigates packet loss during the transition window. An early path switch at the UPF ensures that downlink traffic is redirected to the target before completion of the MOBIKE exchange. A Handover xApp closes the loop by subscribing to KPM telemetry, evaluating UE distribution, and triggering N2 handovers when it detects a load imbalance.

Experimental validation on a four-node testbed demonstrates that the

mechanism operates correctly and preserves user-plane continuity. The measured mean handover latency is approximately 815 ms, with nearly 700 ms attributable to Wi-Fi scanning and re-association rather than to the N2 procedure itself. The MOBIKE exchange contributes only about 7 ms on average, confirming that IPsec tunnel re-binding is not a dominant factor. Target-side buffering reduces packet loss from 54% to 0.4% at 60 Mbps and from 56% to 17% at 100 Mbps, where the remaining loss is caused by temporary Wi-Fi channel saturation during the flush. Outside the handover interval, throughput and jitter remain aligned with baseline performance. The Near-RT RIC control loop itself operates well within its budget: the RC Control round-trip averages 31 ms (P95: 71 ms), and KPM subscriptions are established in under 6 ms.

Disabling IPsec state transfer results in a near doubling of handover latency and progressive instability after repeated mobility events, due to repeated NAS re-registrations and consequent AMF failures. State synchronization is therefore a prerequisite for reliable N3IWF-to-N3IWF mobility, not just a latency optimization.

7.1 Contributions

The central contribution is the integration of the N3IWF as an O-RAN E2 Node. A custom E2SIM agent, compliant with E2AP v3.01 and supporting E2SM-KPM v3 and E2SM-RC v1.03, exposes the N3IWF to the Near-RT RIC as an entity capable of reporting per-UE Wi-Fi metrics and executing RIC-issued control commands. The ASN.1 codec layer was regenerated from scratch to support the current O-RAN specifications, and a metric translation layer converts Wi-Fi station counters into their closest 3GPP DRB-level equivalents. This decouples monitoring and control from the gateway's internal implementation and provides a standardized, access-technology-agnostic interface for managing non-3GPP segments.

A second contribution is the N2-based handover adaptation itself. Since the standard handover model assumes RRC signaling and radio bearer relocation, adapting it to IPsec-based access required redefining the execution semantics around security association continuity and user-plane redirection. The resulting mechanism introduces IKE SA and Child SA state serialization, a MOBIKE-driven execution phase replacing RRC Reconfiguration, and target-side buffering to preserve downlink continuity during tunnel re-binding. Taken together, these components show that reliable non-3GPP mobility depends on explicit IPsec state synchronization rather than on repeated NAS re-registration.

Making this work on top of free5GC required substantial changes to the core network functions. The AMF was modified to support N3IWF target identification, security key refresh during non-3GPP registration updates, and recovery from mid-handover failures. The SMF was extended to perform the early path switch at `HoState_PREPARED`, select the correct UPF endpoint in multi-homed configurations, and clean up residual state after failed attempts.

A proof-of-concept Handover xApp demonstrates end-to-end RIC-driven mobility: it consumes periodic KPM reports, compares per-node UE counts, and dispatches RC Control Requests to rebalance load across N3IWF instances. Although the policy is deliberately simple, it is sufficient to validate the feasibility of closed-loop mobility orchestration over non-3GPP access.

The non-3GPP UE simulator (n3iwue) was also extended to support the handover execution: it deserializes the Target-to-Source container received via IKEv2 Notify, performs the Wi-Fi switch with an on-demand rescan optimization, and executes the MOBIKE address update towards the target N3IWF, with a fallback to full IKE re-establishment if state synchronization is unavailable.

The experimental evaluation also provides a reproducible testbed and a systematic analysis: in particular, the comparison between the MOBIKE path and the full IKE re-establishment fallback shows why IPsec state transfer is necessary to prevent instability under repeated mobility events.

7.2 Limitations

Several limitations constrain the generality and scalability of the results.

The implementation only supports intra-AMF handover. Both N3IWF instances must be registered with the same AMF, as the current free5GC release does not implement inter-AMF UE context transfer. The early path switch introduced at the SMF also deviates from the 3GPP procedure described in TS 23.502 clause 4.9.1.3 [1], where the UPF updates the downlink path only after the Handover Notify. While necessary to preserve user-plane continuity once the UE changes its outer IP address, this modification alters the temporal semantics of the standard flow and would require careful validation in multi-vendor environments.

The dominant contributor to handover latency is the Wi-Fi subsystem, not the N2 signaling. The 300 ms scan delay stems from a workaround for `NetworkManager`'s access point cache behavior, and the $\sim$366 ms reassociation time reflects the baseline IEEE 802.11 procedure on the 2.4 GHz band with no fast-roaming mechanisms (802.11r/k/v) enabled. The mea-

sured latency should therefore not be interpreted as an intrinsic limitation of the handover design but as a characteristic of the access technology in its default configuration. At higher downlink bitrates, temporary channel saturation during the buffer flush introduces residual packet loss that cannot be mitigated at the N3IWF layer.

All experiments were conducted with a single UE and two N3IWF instances in a controlled laboratory setting. The impact of multi-UE contention, higher user densities, heterogeneous traffic profiles, and operation on the 5 GHz or 6 GHz bands remains untested. The computational overhead of IPsec state serialization, buffering, and repeated mobility events was not systematically profiled either.

The xApp's mobility policy relies solely on UE count and picks the first UE in the report list. This is enough for a proof of concept but lacks hysteresis mechanisms, per-UE quality metrics, and anti-ping-pong safeguards that a production deployment would require. The N2 interface in the testbed is also unprotected; in deployment, the IPsec state inside the Source-to-Target Transparent Container would need to be secured through NDS/IP as mandated by TS 33.501 [23].

7.3 Future Directions

Multi-AP N3IWF topology. The current one-to-one mapping between access points and N3IWF instances forces every AP transition through the full N2 handover path. A many-to-one topology, where a single N3IWF manages multiple APs, would allow intra-N3IWF mobility to be resolved locally through 802.11k/v/r or a centralized WLAN controller, reserving N2 handovers for inter-node transitions. This would require per-AP identifiers in the E2 Node abstraction, multi-AP association state at the N3IWF, and RIC-side logic to distinguish local moves from inter-node relocations.

Wi-Fi fast roaming. Integrating IEEE 802.11r (Fast BSS Transition) would eliminate the full WPA2 4-way handshake on each AP switch, potentially reducing re-association to tens of milliseconds. Combined with 802.11k/v for neighbor-report-driven scanning, this could bring the total handover latency well below 200 ms and allow the intrinsic cost of the N2 procedure to be isolated more clearly.

Advanced xApp policies. Replacing the UE-count metric with per-UE throughput, signal quality, or service-level indicators would allow mobility

decisions to reflect user experience rather than load distribution alone. Hysteresis, cooldown timers tied to mobility history, and anti-oscillation guards would bring the policy closer to production-grade Near-RT RIC control strategies.

Inter-AMF and cross-access handover. Once an open-source 5G Core supports inter-AMF context transfer, the mechanism could extend to geographically distributed deployments. A further step would be cross-access handover between gNBs (3GPP) and N3IWFs (non-3GPP) within the same core, coordinating RRC-based and IKEv2-based execution paths under a unified AMF orchestration.

E2SIM scalability. The file-based interface between the N3IWF and the E2SIM is adequate for a small testbed but would not scale to high UE densities. A streaming interface (gRPC or shared memory with change notifications) would reduce polling overhead and improve metric freshness. Systematic profiling of computational and memory overhead under sustained mobility would further clarify the scalability envelope of the architecture.

Source-to-Target container protection. The IPsec state serialized in the Source-to-Target Transparent Container includes keying material (SK_d, SK_ai/ar, SK_ei/er) that travels in cleartext over the N2 interface. While TS 33.501 mandates NDS/IP protection for inter-node signaling, a defense-in-depth approach could encrypt the container end-to-end between source and target N3IWF using a pre-shared or RIC-distributed key, ensuring that even a compromised AMF cannot extract session keys.

Broader E2 use cases. Beyond handover, the E2 integration opens possibilities for other RIC-driven use cases on non-3GPP access. Monitoring of generic Wi-Fi networks through standard KPM xApps, with the advantages of decoupling from the gateway and standardized observation of both Wi-Fi and 5G segments, is a natural extension. Similarly, RIC-driven policy enforcement (e.g., QoS adjustments, access control) through E2SM-RC could be explored without requiring gateway-specific integrations.

Non-cellular KPM extensions. A possible future direction is to define access-technology-specific measurement names within a custom E2SM or to propose extensions to the O-RAN KPM measurement registry that accommodate non-cellular access, reducing the semantic gap between Wi-Fi counters and the existing 3GPP-centric catalogue.

Bibliography

[1] 3GPP, "3GPP TS 23.502: Procedures for the 5G System (5GS)," 3rd Generation Partnership Project, Technical Specification 23.502, 2024, release 18.

[2] ——, "3GPP TS 23.501: System architecture for the 5G System (5GS)," 3rd Generation Partnership Project, Technical Specification 23.501, 2024, release 18.

[3] O-RAN Alliance, "O-RAN Architecture Description," O-RAN Working Group 1, Specification O-RAN.WG1.O-RAN-Architecture-Description-v07.00, 2023.

[4] ——, "O-RAN.WG3.E2SM-KPM: E2 Service Model for Key Performance Measurement," O-RAN Working Group 3, Specification O-RAN.WG3.E2SM-KPM-v03.00, 2024.

[5] ——, "O-RAN.WG3.E2SM-RC: E2 Service Model for RAN Control," O-RAN Working Group 3, Specification O-RAN.WG3.E2SM-RC-v01.03, 2023.

[6] 3GPP, "3GPP TS 24.502: Access to the 5GCN via non-3GPP access networks," 3rd Generation Partnership Project, Technical Specification 24.502, 2024, release 18.

[7] O-RAN Alliance, "O-RAN Alliance," https://www.o-ran.org/, 2024, accessed: January 2026.

[8] ——, "O-RAN.WG3.E2AP: E2 Application Protocol (E2AP)," O-RAN Working Group 3, Specification O-RAN.WG3.E2AP-v03.01, 2024.

[9] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 3rd ed. Pearson, 2024.

[10] E. Dahlman, S. Parkvall, and J. Sköld, *5G NR: The Next Generation Wireless Access Technology*, 2nd ed. Academic Press, 2020.

[11] S. Sesia, I. Toufik, and M. Baker, *LTE - The UMTS Long Term Evolution: From Theory to Practice*, 2nd ed. Wiley, 2011.

[12] 3GPP, "5g system overview," https://www.3gpp.org/technologies/5g-system-overview, 2024, accessed: January 2026.

[13] ——, "About 3gpp," https://www.3gpp.org/about-us, 2024, accessed: January 2026.

[14] ——, "3gpp specification releases," https://www.3gpp.org/specifications/releases, 2024, accessed: January 2026.

[15] IEEE, "IEEE 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," https://www.ieee802.org/11/, 2024, accessed: January 2026.

[16] free5GC Organization, "free5GC: Open Source 5G Core Network," https://free5gc.org/, 2024, accessed: January 2026.

[17] 3GPP, "3GPP TS 29.500: Technical Realization of Service Based Architecture," 3rd Generation Partnership Project, Technical Specification 29.500, 2024, release 18.

[18] O-RAN Alliance, "E2 interface: General Aspects and Principles," O-RAN Working Group 3, Specification O-RAN.WG3.E2GAP-R003-v04.01, 2024.

[19] 3GPP, "3GPP TS 38.413: NG-RAN; NG Application Protocol (NGAP)," 3rd Generation Partnership Project, Technical Specification 38.413, 2024, release 18.

[20] ——, "3GPP TS 38.412: NG-RAN; NG signalling transport," 3rd Generation Partnership Project, Technical Specification 38.412, 2024, release 18.

[21] ——, "3GPP TS 29.281: General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)," 3rd Generation Partnership Project, Technical Specification 29.281, 2023, release 18.

[22] ——, "3GPP TS 29.244: Interface between the Control Plane and the User Plane nodes (PFCP)," 3rd Generation Partnership Project, Technical Specification 29.244, 2023, release 18.

[23] ——, "3GPP TS 33.501: Security architecture and procedures for 5G System," 3rd Generation Partnership Project, Technical Specification 33.501, 2024, release 18.

[24] ——, "3GPP TS 38.300: NR; Overall description; Stage-2," 3rd Generation Partnership Project, Technical Specification 38.300, 2024, release 18.

[25] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, "Extensible authentication protocol (eap)," RFC 3748, Jun. 2004. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3748

[26] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen, "RFC 7296: Internet Key Exchange Protocol Version 2 (IKEv2)," IETF, RFC 7296, 2014.

[27] P. Eronen, "RFC 4555: IKEv2 Mobility and Multihoming Protocol (MOBIKE)," IETF, RFC 4555, 2006.

[28] S. Kent and K. Seo, "RFC 4301: Security Architecture for the Internet Protocol (IPsec)," IETF, RFC 4301, 2005.

[29] S. Kent, "RFC 4303: IP Encapsulating Security Payload (ESP)," IETF, RFC 4303, 2005.

[30] 3GPP, "3GPP TS 28.552: Management and orchestration; 5G performance measurements," 3rd Generation Partnership Project, Technical Specification 28.552, 2024, release 18.

[31] A. Feraudo, S. Maxenti, A. Lacava, P. Bellavista, M. Polese, and T. Melodia, "xdevsm: Streamlining xapp development with a flexible framework for o-ran e2 service models," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, ser. ACM MobiCom '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 1954–1961. [Online]. Available: https://doi.org/10.1145/3636534.3697325

[32] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[33] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open, programmable, and virtualized 5G networks: State-of-the-art and the road ahead," *Computer Networks*, vol. 182, p. 107516, 2020.

[34] S. D'Oro, L. Bonati, M. Polese, and T. Melodia, "OrchestRAN: Network automation through orchestrated intelligence in the open RAN," in *IEEE INFOCOM 2022*. IEEE, 2022, pp. 270–279.

[35] A. Lacava, M. Polese, R. Sivaraj, R. Soundararajan, B. S. Bhatt, T. Singh, T. Zugno, F. Cuomo, and T. Melodia, "Programmable and customized intelligence for traffic steering in 5G networks using open RAN architectures," *IEEE Transactions on Mobile Computing*, vol. 23, no. 4, pp. 2882–2897, 2024.

[36] S. Marinova and A. Leon-Garcia, "Intelligent O-RAN beyond 5G: Architecture, use cases, challenges, and opportunities," *IEEE Access*, vol. 12, pp. 54 982–55 008, 2024.

[37] A. Kunz and A. K. Salkintzis, "Non-3GPP access security in 5G," *Journal of ICT Standardization*, vol. 8, no. 1, pp. 49–68, 2020.

[38] M. T. Lemes, A. M. Alberti, C. B. Both, and A. Cardoso Júnior, "A tutorial on trusted and untrusted non-3GPP accesses in 5G systems—first steps toward a unified communications infrastructure," in *IEEE Latin-American Conference on Communications (LATINCOM)*. IEEE, 2022, pp. 1–6.

[39] H.-W. Wu, S. Ferlin, G. Caso, M. Özger, Ö. Alay, and A. Brunstrom, "A survey on multipath transport protocols towards 5G access traffic steering, switching and splitting," *IEEE Access*, vol. 9, pp. 164 417–164 439, 2021.

[40] 3GPP, "Study on access traffic steering, switch and splitting support in the 5G system architecture," 3rd Generation Partnership Project, Technical Report TR 23.793, 2019, release 16.

[41] J. Prados-Garzon, P. Ameigeiras, J. J. Ramos-Muñoz, P. Andres-Maldonado, and J. M. Lopez-Soler, "5G non-public networks: Standardization, architectures and challenges," *IEEE Access*, vol. 9, pp. 153 893–153 908, 2021.

[42] F. Zampognaro, D. Verde, and M. Luglio, "Local edge-services through N3IWF in 5G networks," in *2024 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*. IEEE, 2024, pp. 1–6.

[43] T. Mukute, L. Mamushiane, A. A. Lysko, and E. R. Modroiu, "Control plane performance benchmarking and feature analysis of popular open-source 5G core networks: OpenAirInterface, open5gs, and free5GC," *IEEE Access*, vol. 12, pp. 65 411–65 429, 2024.

[44] F. Bashar, A. Tripathi, M. R. Chowdhury, and A. Da Silva, "Inter-DU load balancing in an experimental over-the-air 5G open radio access network," *arXiv preprint arXiv:2412.21161*, 2025.

[45] A. Vicario, A. Pagano, A. Dino, and D. Croce, "Towards intelligent mobility management: Customized handover in 5G O-RAN networks," in *Proceedings of the ACM*. ACM, 2025.

[46] M. Barbosa and K. Dias, "Open RAN-enabled deep learning-assisted mobility management for connected vehicles," *arXiv preprint arXiv:2412.21161*, 2024.

[47] A. Haghrah, M. P. Abdollahi, and H. Azarhava, "A survey on the handover management in 5G-NR cellular networks: Aspects, approaches and challenges," *EURASIP Journal on Wireless Communications and Networking*, vol. 2023, p. 52, 2023.

[48] Y. Ullah, M. B. Roslee, S. M. Mitani, S. A. Khan, and M. H. Jusoh, "A survey on handover and mobility management in 5G HetNets: Current state, challenges, and future directions," *Sensors*, vol. 23, no. 11, p. 5081, 2023.

[49] A. I. Manolopoulos and V.-M. Alevizaki, "Demonstration of multi-access 6G networks with user mobility considerations," in *2024 IEEE 29th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE, 2024, pp. 1–6.

[50] O-RAN Software Community, "O-RAN SC E2 Simulator (e2sim)," https://github.com/o-ran-sc/sim-e2-interface, 2024, accessed: January 2026.

[51] "FlexRIC: Flexible O-RAN RIC Platform," https://github.com/openaicellular/flexric, 2024, accessed: January 2026.

[52] M. Pala, "ASN.1 Compiler (mouse07410 fork)," https://github.com/mouse07410/asn1c, 2024, accessed: January 2026.

[53] O-RAN Software Community, "RSAC Decision: ASN.1 Compiler Recommendation," https://wiki.o-ran-sc.org/display/RSAC/ASN.1+Compiler, 2025, accessed: January 2026.

[54] "xDevSM Tutorial," https://openrangym.com/tutorials/xdevsm-tutorial, 2024, accessed: January 2026.

[55] free5GC Organization, "n3iwue: Non-3GPP UE Simulator," https://github.com/free5gc/n3iwue, 2024, accessed: January 2026.

Appendix A

This appendix provides supplementary code listings, configuration excerpts, and data structures referenced throughout the thesis.

A.1 Data Structures and Interface Formats


```
{
  "timestamp": "2026-02-09T17:07:53.825340067Z",
  "associations": [
    {
      "interface": "wlp44s0",
      "mac": "4c:b0:4a:9e:27:94",
      "ueIp": "192.168.11.50",
      "station": {
        "interface": "wlp44s0",
        "mac": "4c:b0:4a:9e:27:94",
        "fields": {
          "signal": "-23",
          "signal_avg": "-23 dBm",
          "iw.signal": "-23 [-23, -26] dBm",
          "iw.signal_avg": "-23 dBm",
          "rx_bytes": "10912017",
          "iw.rx_bytes": "10912017",
          "tx_bytes": "630558666",
          "iw.tx_bytes": "630558666",
          "rx_packets": "64477",
          "iw.rx_packets": "64477",
          "tx_packets": "577530",
          "iw.tx_packets": "577530",
          "inactive_msec": "6924",
          "iw.inactive_time": "6927 ms",
          "connected_time": "26395",
          "iw.connected_time": "26395 seconds",
          "iw.rx_bitrate": "130.0 MBit/s MCS 15",
          "iw.tx_bitrate": "130.0 MBit/s MCS 15",
```

```
        "iw.rx_drop_misc": "2",
        "iw.tx_failed": "0",
        "iw.tx_retries": "412"
      }
    },
    "ue": {
      "ranUeNgapId": 1,
      "amfUeNgapId": 2
    }
  }
 ]
}
```

Listing A.1: Structure of the rc_metrics.json file written by the N3IWF and consumed by the E2SIM for KPM report generation

```
struct RcUeInfoSnapshot {
    int64_t ran_ue_ngap_id{-1};
    int64_t amf_ue_ngap_id{-1};
    std::string ip_addr_v4;
    std::string guti;
    std::vector<RcPduSessionSnapshot> pdu_sessions;
    // ...
};

struct RcAssociationSnapshot {
    std::string interface_name;
    std::string mac;
    std::string ue_ip;
    RcStationSnapshot station;
    RcUeInfoSnapshot ue;
};
```

Listing A.2: E2SIM-side data structures for N3IWF integration (snapshot of UE info and station metrics)

```
{
  "ranUeNgapId": 12345,
  "directForwarding": false,
  "targetId": "208-93-136",
  "metadata": {
    "ueIdentity": "gNB-UEID ran=...",
    "targetMcc": "208",
    "targetMnc": "93",
    "targetNrCellId": "136"
  }
}
```

Listing A.3: JSON payload sent by the E2SIM to the N3IWF HTTP endpoint to trigger an N2 handover

```
type IKESAState struct {
    LocalSPI uint64 `json:"localSpi"`
    RemoteSPI uint64 `json:"remoteSpi"`
    InitiatorMessageID uint32 `json:"initiatorMsgId"`
    ResponderMessageID uint32 `json:"responderMsgId"`
    State           uint8 `json:"state"`
    ConcatenatedNonce string `json:"concatenatedNonce,omitempty"`
    // Cryptographic algorithm negotiation
    Encr TransformJSON `json:"encr"`
    Integ TransformJSON `json:"integ"`
    Prf  TransformJSON `json:"prf"`
    Dh   TransformJSON `json:"dh"`
    // Derived keying material (base64)
    SK_d string `json:"sk_d,omitempty"`
    SK_ai string `json:"sk_ai,omitempty"`
    SK_ar string `json:"sk_ar,omitempty"`
    SK_ei string `json:"sk_ei,omitempty"`
    SK_er string `json:"sk_er,omitempty"`
    SK_pi string `json:"sk_pi,omitempty"`
    SK_pr string `json:"sk_pr,omitempty"`
    // NAT and MOBIKE state
    UeBehindNAT  bool `json:"ueBehindNat,omitempty"`
    N3iwfBehindNAT bool `json:"n3iwfBehindNat,omitempty"`
    MobikeEnabled bool `json:"mobikeEnabled,omitempty"`
}

type ChildSAState struct {
    InboundSPI uint32 `json:"inboundSpi"`
    OutboundSPI uint32 `json:"outboundSpi"`
    // SA role and protocol
    LocalIsInitiator bool `json:"localIsInitiator"`
    SelectedIPProto uint8 `json:"selectedIpProto"`
    PeerPublicIP string `json:"peerPublicIp"`
    TrafficSelectorLocal string `json:"tsLocal"`
    TrafficSelectorRemote string `json:"tsRemote"`
    // UDP encapsulation (NAT traversal)
    EnableEncapsulate bool `json:"enableEncapsulate"`
    N3IWFPort       int `json:"n3iwfPort,omitempty"`
    NATPort         int `json:"natPort,omitempty"`
    PDUSessionIds []int64 `json:"pduSessionIds,omitempty"`
    XfrmiId uint32 `json:"xfrmiId"`
    // Negotiated transforms
    Encr TransformJSON `json:"encr"`
    Integ TransformJSON `json:"integ"`
```

```
    ESN  TransformJSON `json:"esn"`
    // XFRM replay state (anti-replay counters)
    InboundReplay *xfrm.ReplayState `json:"inboundReplay,omitempty"`
    OutboundReplay *xfrm.ReplayState `json:"outboundReplay,omitempty"`
    // Directional keying material (base64)
    InitiatorToResponderEncryptionKey string `json:"i2rEncrKey,omitempty
        "`
    ResponderToInitiatorEncryptionKey string `json:"r2iEncrKey,omitempty
        "`
    InitiatorToResponderIntegrityKey string `json:"i2rIntegKey,omitempty
        "`
    ResponderToInitiatorIntegrityKey string `json:"r2iIntegKey,omitempty
        "`
}
```

Listing A.4: IPsec state transfer data structures (from `internal/handover/statesync/`)

A.2 xApp and E2SIM Code

Listing A.5 shows the core logic of the Handover xApp: the indication handler that accumulates KPM reports, the source/target selection, and the RC control request dispatch (see Section 5.2 for discussion).

```
class xAppMonControlContainer():

    def ind_msg_handler(self, ind_hdr, ind_msg, meid):
        gnbid = meid.decode('utf-8')
        self.ind_count_by_meid[gnbid] += 1

        if ind_msg.type.value == FORMAT_3_INDICATION_MESSAGE:
            ue_count = int(ind_msg.data.frm_3.ue_meas_report_lst_len)
            self.last_ue_count_by_meid[gnbid] = ue_count

        if self.handover_sent:
            return

        if self.pending_handover is None:
            if self.ind_count_by_meid.get(gnbid, 0) >= self.threshold:
                source, target = self._select_source_target()
                if source and target:
                    self.pending_handover = (source, target)
                    self._try_send_handover()

        if self.pending_handover is not None:
            self._try_send_handover()
```

```
23
24      def _select_source_target(self):
25          meids = [meid for meid in self.subscribed_meids
26                   if meid in self.ind_count_by_meid]
27          if len(meids) < 2:
28              return None, None
29
30          def sort_key(meid: str):
31              return (self._get_load_metric(meid),
32                      self.ind_count_by_meid.get(meid, 0), meid)
33
34          source = max(meids, key=sort_key)
35          targets = [m for m in meids if m != source]
36          if not targets:
37              return None, None
38          target = min(targets, key=sort_key)
39          return source, target
40
41      def _try_send_handover(self) -> bool:
42          if self.handover_sent or self.pending_handover is None:
43              return False
44          source_meid, target_meid = self.pending_handover
45
46          ue_struct = self.last_ue_struct_by_meid.get(source_meid)
47          if ue_struct is None:
48              return False
49
50          rc_desc = self.rc_func_desc_by_meid.get(source_meid)
51          target_info = self.gnb_info_by_meid.get(target_meid, {})
52          self.rc_func.set_nr_cell_id(target_info["globalNbId"]["nbId"])
53          self.rc_func.set_plmn_identity(target_info["globalNbId"]["plmnId
                "])
54          self.rc_func.send(
55              e2_node_id=source_meid,
56              ran_func_dsc=rc_desc,
57              ue_id_struct=ue_struct,
58              control_action_id=1)
59          self._reset_all_counters()
60          self.handover_sent = True
61          return True
```

Listing A.5: Handover xApp: indication handler, load evaluation, and RC control dispatch

Listing A.6 shows the E2SIM-side function that translates an RC Control Request into an HTTP POST to the N3IWF (see Section 5.1.5).

```
1 static N3iwfTriggerResponse trigger_n3iwf_handover(
2     const RcControlContext &ctx,
```

```
    const std::string &command_desc)
{
  json payload;
  build_handover_request_payload(ctx, payload, payload_error);

  std::string url = handover_endpoint_url();

  http_post_json(url, payload.dump(),
                 http_code, http_body, curl_error);

  // Parse response and return success/failure
}
```

Listing A.6: E2SIM: trigger handover via HTTP POST to N3IWF

```
#!/usr/bin/env python3
import os, time, json, subprocess, re

HOSTAPD_CTRL = os.environ.get("HOSTAPD_CTRL_PATH", "/var/run/hostapd")
IFACES = os.environ.get("HOSTAPD_IFACES", "wlp6s0").split()
SCRAPE_INTERVAL = float(os.environ.get("SCRAPE_INTERVAL", "1"))
OUTDIR = os.environ.get("OUTPUT_DIR", "/var/run/wifi-metrics")

def hostapd_all_sta(iface):
    txt = run(f"hostapd_cli -p {HOSTAPD_CTRL} -i {iface} all_sta")
    stas, cur = {}, None
    for line in txt.splitlines():
        line = line.strip()
        if re.match(r"^[0-9a-f]{2}(:[0-9a-f]{2}){5}$", line):
            cur = line
            stas[cur] = {}
        elif cur and "=" in line:
            k, v = line.split("=", 1)
            stas[cur][k] = v
    return stas

def iw_station_dump(iface):
    txt = run(f"iw dev {iface} station dump", timeout=5)
    blocks = re.split(r"\n(?=Station\s)", txt)
    res = {}
    for b in blocks:
        m = re.search(r"Station\s([0-9a-f:]{17})", b)
        if not m: continue
        mac = m.group(1)
        data = {}
        for line in b.splitlines():
            kv = line.strip().split(":", 1)
            if len(kv) == 2:
```

```
                k = kv[0].strip().lower().replace(" ", "_")
                data[k] = kv[1].strip()
        res[mac] = data
    return res

def main():
    while True:
        payload = {}
        for iface in IFACES:
            entry = {"ts": int(time.time())}
            entry["hostapd"] = {"stations": hostapd_all_sta(iface)}
            entry["station_dump"] = iw_station_dump(iface)
            payload[iface] = entry
        write_atomic(os.path.join(OUTDIR, "metrics.json"),
                     json.dumps(payload, indent=2).encode())
        time.sleep(SCRAPE_INTERVAL)
```

Listing A.7: Wi-Fi Metrics Exporter (runs on each AP host)

A.3 ASN.1 Excerpts

The following excerpts are taken from the ASN.1 source files used to regenerate the E2SIM codec layer (Section 5.1.1). They show the key information elements for the two E2 Service Models used in this thesis.

A.3.1 E2SM-KPM v3 – Indication Message

Listing A.8 shows the top-level KPM Indication Message and the Format 3 (per-UE report list) used by the E2SIM to deliver Wi-Fi metrics to the xApp. Each `UEMeasurementReportItem` contains a `UEID` and a Format 1 measurement record carrying the actual metric values.

```
E2SM-KPM-IndicationMessage ::= SEQUENCE {
    indicationMessage-formats CHOICE {
        indicationMessage-Format1 E2SM-KPM-IndicationMessage-Format1,
        indicationMessage-Format2 E2SM-KPM-IndicationMessage-Format2,
        indicationMessage-Format3 E2SM-KPM-IndicationMessage-Format3,
        ...
    },
    ...
}

E2SM-KPM-IndicationMessage-Format1 ::= SEQUENCE {
    measData      MeasurementData,
    measInfoList  MeasurementInfoList OPTIONAL,
```

```
    granulPeriod  GranularityPeriod   OPTIONAL,
    ...
}

E2SM-KPM-IndicationMessage-Format3 ::= SEQUENCE {
    ueMeasReportList UEMeasurementReportList,
    ...
}

UEMeasurementReportList ::= SEQUENCE (SIZE(1..maxnoofUEMeasReport))
    OF UEMeasurementReportItem

UEMeasurementReportItem ::= SEQUENCE {
    ueID      UEID,
    measReport E2SM-KPM-IndicationMessage-Format1,
    ...
}
```

Listing A.8: E2SM-KPM v3 ASN.1: IndicationMessage and Format 3 (from `e2sm_kpm_v03.00_modified.asn1`)

A.3.2 E2SM-RC v1.03 – Control Header and Message

Listing A.9 shows the RC Control Header (Format 1, which identifies the target UE and the control style/action) and the Control Message (Format 1, which carries the RAN parameters, in this thesis the target cell identity for handover).

```
E2SM-RC-ControlHeader ::= SEQUENCE {
    ric-controlHeader-formats CHOICE {
        controlHeader-Format1 E2SM-RC-ControlHeader-Format1,
        controlHeader-Format2 E2SM-RC-ControlHeader-Format2,
        ...
    },
    ...
}

E2SM-RC-ControlHeader-Format1 ::= SEQUENCE {
    ueID                UEID,
    ric-Style-Type      RIC-Style-Type,
    ric-ControlAction-ID RIC-ControlAction-ID,
    ric-ControlDecision ENUMERATED {accept, reject, ...} OPTIONAL,
    ...
}

E2SM-RC-ControlMessage ::= SEQUENCE {
    ric-controlMessage-formats CHOICE {
```

```
        controlMessage-Format1 E2SM-RC-ControlMessage-Format1,
        controlMessage-Format2 E2SM-RC-ControlMessage-Format2,
        ...
    },
    ...
}

E2SM-RC-ControlMessage-Format1 ::= SEQUENCE {
    ranP-List SEQUENCE (SIZE(0..maxnoofAssociatedRANParameters))
        OF E2SM-RC-ControlMessage-Format1-Item,
    ...
}

E2SM-RC-ControlMessage-Format1-Item ::= SEQUENCE {
    ranParameter-ID      RANParameter-ID,
    ranParameter-valueType RANParameter-ValueType,
    ...
}
```

Listing A.9: E2SM-RC v1.03 ASN.1: ControlHeader and ControlMessage (from `e2sm_rc_v1_03_modified.asn`)

A.4 Testbed Configuration

This section provides the full configuration files used on the Dell Edge Gateway 5200 node (5GC, N3IWF-A, E2Node-A).

A.4.1 Docker Compose (Dell Edge Gateway 5200)

Listing A.10 shows the Docker Compose file (`dcb.yaml`) that deploys the entire 5GC, N3IWF-A, E2Node-A, and the Wi-Fi metrics exporter on Dell Edge Gateway 5200.

```
services:
  free5gc-upf:
    container_name: upf
    build:
      context: ./nf_upf
      args:
        DEBUG_TOOLS: "false"
    command: bash -c "./upf-iptables.sh && iperf3 -s -D && ./upf -c ./
        config/upfcfg.yaml"
    ports:
      - "2152:2152/udp"
      - "8805:8805/udp"
```

```
    - "5202:5201"
    - "5202:5201/udp"
  volumes:
    - ./config/upfcfg.yaml:/free5gc/config/upfcfg.yaml
    - ./config/upf-iptables.sh:/free5gc/upf-iptables.sh
  cap_add:
    - NET_ADMIN
  networks:
    privnet:
      ipv4_address: 10.100.200.17

db:
  container_name: mongodb
  image: mongo:3.6.8
  command: mongod --port 27017
  networks:
    privnet:
      ipv4_address: 10.100.200.2

free5gc-nrf:
  container_name: nrf
  build: { context: ./nf_nrf }
  command: ./nrf -c ./config/nrfcfg.yaml
  volumes:
    - ./config/nrfcfg.yaml:/free5gc/config/nrfcfg.yaml
    - ./cert:/free5gc/cert
  environment:
    DB_URI: mongodb://db/free5gc
  networks:
    privnet:
      ipv4_address: 10.100.200.3
  depends_on: [db]

free5gc-amf:
  container_name: amf
  image: zhria/amfcustom:latest
  command: ./amf -c ./config/amfcfg.yaml
  ports:
    - "8000:8000"
    - "38412:38412/sctp"
  volumes:
    - ./config/amfcfg.yaml:/free5gc/config/amfcfg.yaml
    - ./cert:/free5gc/cert
  cap_add: [NET_ADMIN]
  networks:
    privnet:
      ipv4_address: 10.100.200.16
  depends_on: [free5gc-nrf]

```

```
61  free5gc-smf:
62    container_name: smf
63    image: zhria/smfcustom:latest
64    command: ./smf -c ./config/smfcfg.yaml -u ./config/uerouting.yaml
65    volumes:
66      - ./config/smfcfg.yaml:/free5gc/config/smfcfg.yaml
67      - ./config/uerouting.yaml:/free5gc/config/uerouting.yaml
68      - ./cert:/free5gc/cert
69    networks:
70      privnet:
71        ipv4_address: 10.100.200.7
72    depends_on: [free5gc-nrf, free5gc-upf]
73
74  free5gc-n3iwf:
75    container_name: n3iwf
76    image: zhria/n3iwfcustom:latest
77    command: /free5gc/n3iwf/n3iwf -c /free5gc/config/n3iwfcfg.yaml
78    volumes:
79      - ./config/n3iwfcfg.yaml:/free5gc/config/n3iwfcfg.yaml
80      - ./n3iwf_e2logger:/free5gc/log/:rw
81      - ./wifi_metrics:/free5gc/wifi_metrics:ro
82    cap_add: [NET_ADMIN]
83    networks:
84      privnet:
85        ipv4_address: 10.100.200.15
86      macvlan10:
87        ipv4_address: 192.168.10.254
88    depends_on: [free5gc-amf, free5gc-smf, free5gc-upf]
89
90  free5gc-e2node:
91    container_name: e2node
92    image: zhria/e2node:latest
93    command: ./build/kpm_callbacks -c ./config/e2node.yaml
94    network_mode: "host"
95    volumes:
96      - ./n3iwf_e2logger:/home/e2sim/log/:rw
97      - ./config/e2node.yaml:/home/e2sim/config/e2node.yaml
98    depends_on: [free5gc-n3iwf]
99
100 wifi-metrics-exporter:
101   image: alpine:3.19
102   container_name: wifi-metrics-exporter
103   network_mode: host
104   pid: host
105   privileged: true
106   command: >-
107     sh -lc 'apk add --no-cache python3 iw iproute2 hostapd &&
108     python3 /app/exporter.py'
109   volumes:
```

```
      - ./wifi_metrics/exporter.py:/app/exporter.py:ro
      - ./wifi_metrics:/wifi-metrics:rw
      - /run/hostapd:/var/run/hostapd:rw
    environment:
      HOSTAPD_CTRL_PATH: /var/run/hostapd
      HOSTAPD_IFACES: "wlp6s0"
      SCRAPE_INTERVAL: "0.5"
      OUTPUT_DIR: "/wifi-metrics"

networks:
  privnet:
    ipam:
      config:
        - subnet: 10.100.200.0/24
    driver_opts:
      com.docker.network.bridge.name: br-free5gc
  macvlan10:
    external: true

volumes:
  dbdata:
```

Listing A.10: Docker Compose deployment on Dell Edge Gateway 5200 (dcb.yaml)

Note: Supporting NFs (AUSF, UDM, UDR, NSSF, PCF, NEF, CHF, WebUI) are omitted for brevity; they follow the same pattern (build from local context, mount config YAML, join ***privnet****).*

A.4.2 N3IWF Configuration (Dell Edge Gateway 5200 – N3IWF-A)

```
info:
  version: 1.0.5
  description: N3IWF initial local configuration

configuration:
  n3iwfInformation:
    globalN3IWFID:
      plmnID:
        mcc: 208
        mnc: 93
      n3iwfID: 135
    name: free5GC_N3IWF
    supportedTAList:
      - tac: 000001
        broadcastPlmnList:
```

```
        - plmnID: { mcc: 208, mnc: 93 }
          taiSliceSupportList:
            - snssai: { sst: 1, sd: 010203 }
            - snssai: { sst: 1, sd: 112233 }

  amfSCTPAddresses:
    - ip: [amf.free5gc.org]
      port: 38412
  nasTcpPort: 20000

  ikeBindAddress: 192.168.10.254
  ipSecTunnelAddress: 10.0.0.1
  ueIpAddressRange: 10.0.0.0/24
  xfrmInterfaceName: xfrmi
  xfrmInterfaceID: 1
  xfrmMTU: 1380

  n3iwfGtpBindAddress: 10.100.200.15

  wifi:
    ssid: N3IWF_AP-A
    password: MMWEdge3987#!

  fqdn: n3iwf.free5gc.org
  privateKey: cert/n3iwf.key
  certificateAuthority: cert/n3iwf.pem
  certificate: cert/n3iwf.pem

  livenessCheck:
    enable: true
    transFreq: 60s
    maxRetryTimes: 4

  handoverStateSync:
    enable: true

logger:
  enable: true
  level: info
```

Listing A.11: N3IWF-A configuration (n3iwfcfg.yaml)

A.4.3 AMF Configuration

```
info:
  version: 1.0.9
  description: AMF initial local configuration
```

```yaml

configuration:
  amfName: AMF
  ngapIpList: [0.0.0.0]
  ngapPort: 38412

  sbi:
    scheme: http
    registerIPv4: amf.free5gc.org
    bindingIPv4: 0.0.0.0
    port: 8000

  serviceNameList: [namf-comm, namf-evts, namf-mt, namf-loc, namf-oam]

  servedGuamiList:
    - plmnId: { mcc: 208, mnc: 93 }
      amfId: cafe00

  supportTaiList:
    - plmnId: { mcc: 208, mnc: 93 }
      tac: 000001

  plmnSupportList:
    - plmnId: { mcc: 208, mnc: 93 }
      snssaiList:
        - sst: 1
          sd: 010203
        - sst: 1
          sd: 112233

  supportDnnList: [internet]
  nrfUri: http://nrf.free5gc.org:8000

  security:
    integrityOrder: [NIA2]
    cipheringOrder: [NEA0, NEA2]

  sctp:
    numOstreams: 3
    maxInstreams: 5
    maxAttempts: 2
    maxInitTimeout: 2

logger:
  enable: true
  level: info
```

Listing A.12: AMF configuration (amfcfg.yaml)

A.4.4 SMF Configuration

```
info:
  version: 1.0.7
  description: SMF initial local configuration

configuration:
  smfName: SMF

  sbi:
    scheme: http
    registerIPv4: 10.100.200.7
    bindingIPv4: 10.100.200.7
    port: 8000

  serviceNameList: [nsmf-pdusession, nsmf-event-exposure, nsmf-oam]

  snssaiInfos:
    - sNssai: { sst: 1, sd: 010203 }
      dnnInfos:
        - dnn: internet
          dns: { ipv4: 8.8.8.8 }
    - sNssai: { sst: 1, sd: 112233 }
      dnnInfos:
        - dnn: internet
          dns: { ipv4: 8.8.8.8 }

  pfcp:
    nodeID: 10.100.200.7
    listenAddr: 10.100.200.7
    externalAddr: 10.100.200.7
    heartbeatInterval: 10s

  userplaneInformation:
    upNodes:
      gNB1:
        type: AN
      UPF:
        type: UPF
        nodeID: 10.100.200.17
        addr: 10.100.200.17
        sNssaiUpfInfos:
          - sNssai: { sst: 1, sd: 010203 }
            dnnUpfInfoList:
              - dnn: internet
                pools: [{ cidr: 10.60.0.0/16 }]
          - sNssai: { sst: 1, sd: 112233 }
            dnnUpfInfoList:
```

```
            - dnn: internet
              pools: [{ cidr: 10.61.0.0/16 }]
      interfaces:
        - interfaceType: N3
          endpoints: [10.100.200.17, 192.168.17.20]
          networkInstances: [internet]
  links:
    - A: gNB1
      B: UPF

nrfUri: http://10.100.200.3:8000
urrPeriod: 30
urrThreshold: 100000000

logger:
  enable: true
  level: warn
```

Listing A.13: SMF configuration (smfcfg.yaml)

A.4.5 UPF Configuration

```
version: 1.0.3
description: UPF initial local configuration

pfcp:
  addr: 10.100.200.17
  nodeID: 10.100.200.17
  retransTimeout: 4s
  maxRetrans: 5

gtpu:
  forwarder: gtp5g
  ifList:
    - addr: 0.0.0.0
      type: N3
      mtu: 1380

dnnList:
  - dnn: internet
    cidr: 10.60.0.0/16
  - dnn: internet
    cidr: 10.61.0.0/16

logger:
  enable: true
  level: trace
```

Listing A.14: UPF configuration (upfcfg.yaml)

A.4.6 E2SIM Configuration (Dell Edge Gateway 5200 – E2Node-A)

```
info:
  version: 1.0.0
  description: E2Node initial configuration

configuration:
  e2NodeName: E2Node
  logging:
    file: "/home/e2sim/log/e2node.log"
    consoleLevel: "warn"
    teeHighToFile: false
  ricAddress: [192.168.17.74]
  ricPort: 32222
  localAddress: [192.168.17.20]
  n3iwfHandoverUrl: "http://10.100.200.15:9085/rc/handover"
  gnbCuUpId: 1
  gnbDuId: 1
```

Listing A.15: E2SIM configuration (e2node.yaml)

A.4.7 UP Xtreme i12 Edge 1 Configuration (N3IWF-B, E2Node-B)

Unlike Dell Edge Gateway 5200, UP Xtreme i12 Edge 1 does not host 5GC NFs. All three containers run with `network_mode: host`, because the N3IWF-B reaches the AMF and UPF over the Ethernet subnet (`192.168.17.0/24`) rather than through a local Docker bridge.

```
services:
  free5gc-n3iwf:
    container_name: n3iwf
    image: zhria/n3iwfcustom:latest
    command: /free5gc/n3iwf/n3iwf -c /free5gc/config/n3iwfcfg.yaml
    network_mode: host
    volumes:
      - ./config/n3iwfcfg.yaml:/free5gc/config/n3iwfcfg.yaml
      - ./config/n3iwf-ipsec.sh:/free5gc/n3iwf-ipsec.sh
      - ./cert/n3iwf.key:/free5gc/cert/n3iwf.key
      - ./cert/n3iwf.pem:/free5gc/cert/n3iwf.pem
```

```
      - ./n3iwf_e2logger:/free5gc/log/:rw
      - ./wifi_metrics:/free5gc/wifi_metrics:ro
    environment:
      GIN_MODE: release
    cap_add: [NET_ADMIN]

  free5gc-e2node:
    container_name: e2node
    image: zhria/e2node:latest
    command: ./build/kpm_callbacks -c ./config/e2node.yaml
    network_mode: host
    volumes:
      - ./n3iwf_e2logger:/home/e2sim/log/:rw
      - ./config/e2node.yaml:/home/e2sim/config/e2node.yaml
    depends_on: [free5gc-n3iwf]

  wifi-metrics-exporter:
    image: alpine:3.19
    container_name: wifi-metrics-exporter
    network_mode: host
    pid: host
    privileged: true
    command: >-
      sh -lc 'apk add --no-cache python3 iw iproute2 hostapd &&
      python3 /app/exporter.py'
    volumes:
      - ./wifi_metrics/exporter.py:/app/exporter.py:ro
      - ./wifi_metrics:/wifi-metrics:rw
      - /run/hostapd:/var/run/hostapd:rw
    environment:
      HOSTAPD_CTRL_PATH: /var/run/hostapd
      HOSTAPD_IFACES: "wlp44s0"
      SCRAPE_INTERVAL: "0.5"
      OUTPUT_DIR: "/wifi-metrics"
```

Listing A.16: Docker Compose deployment on UP Xtreme i12 Edge 1 (dcb.yaml)

```
info:
  version: 1.0.5
  description: N3IWF initial local configuration

configuration:
  n3iwfInformation:
    globalN3IWFID:
      plmnID:
        mcc: 208
        mnc: 93
```

```
    n3iwfID: 136
  name: free5GC_N3IWF
  supportedTAList:
    - tac: 000001
      broadcastPlmnList:
        - plmnID: { mcc: 208, mnc: 93 }
          taiSliceSupportList:
            - snssai: { sst: 1, sd: 010203 }
            - snssai: { sst: 1, sd: 112233 }

amfSCTPAddresses:
  - ip: [192.168.17.20]
    port: 38412
nasTcpPort: 20000
localSctpAddr: 192.168.17.181

ikeBindAddress: 192.168.11.1
ipSecTunnelAddress: 10.0.0.1
ueIpAddressRange: 10.0.0.0/24
xfrmInterfaceName: xfrmi
xfrmInterfaceID: 1
xfrmMTU: 1380

n3iwfGtpBindAddress: 192.168.17.181

wifi:
  ssid: N3IWF_AP-B
  password: TestWifiZakHandoverN3IWF2

fqdn: n3iwf.free5gc.org
privateKey: cert/n3iwf.key
certificateAuthority: cert/n3iwf.pem
certificate: cert/n3iwf.pem

livenessCheck:
  enable: true
  transFreq: 60s
  maxRetryTimes: 4

handoverStateSync:
  enable: true

hoBuffer:
  enabled: true

logger:
  enable: true
  level: debug
```

Listing A.17: N3IWF-B configuration on UP Xtreme i12 Edge 1 (n3iwfcfg.yaml)

```yaml
info:
  version: 1.0.0
  description: E2Node initial configuration

configuration:
  e2NodeName: E2Node
  ricAddress: [192.168.17.74]
  ricPort: 32222
  localAddress: [192.168.17.181]
  n3iwfHandoverUrl: "http://192.168.17.181:9085/rc/handover"
  gnbCuUpId: 2
  gnbDuId: 2
```

Listing A.18: E2SIM configuration on UP Xtreme i12 Edge 1 (e2node.yaml)

A.4.8 Handover xApp Descriptor

```json
{
    "name": "ho-control-xapp",
    "version": "0.0.118",
    "appInstanceName": "ho-control-xapp",
    "containers": [
        {
            "name": "ho-control-xapp",
            "image": {
                "registry": "docker.io",
                "name": "zhria/handover_app",
                "tag": "0.0.118"
            },
            "imagePullPolicy": "Always"
        }
    ],
    "messaging": {
        "ports": [
            {
                "name": "http",
                "container": "ho-control-xapp",
                "port": 8080,
                "description": "http service"
            },
            {
                "name": "rmrroute",
```

```
                "container": "ho-control-xapp",
                "port": 4561,
                "description": "rc route port"
            },
            {
                "name": "rmrdata",
                "container": "ho-control-xapp",
                "port": 4560,
                "rxMessages": [
                    "RIC_SUB_RESP",
                    "RIC_SUB_FAILURE",
                    "RIC_SUB_DEL_RESP",
                    "RIC_CONTROL_ACK",
                    "RIC_CONTROL_FAILURE"
                ],
                "txMessages": [
                    "RIC_SUB_REQ",
                    "RIC_SUB_DEL_REQ",
                    "RIC_CONTROL_REQ"
                ],
                "policies": [1],
                "description": "rmr data port for rc-xapp"
            }
        ]
    },
    "controls": {
        "subscription": {
            "url": "http://service-ricxapp-ho-control-xapp-http:8080/",
            "clientEndpoint": "service-ricxapp-ho-control-xapp-http"
        }
    }
}
```

Listing A.19: Handover xApp descriptor (config-file.json)