Towards Intelligent Mobility Management: Customized Handover in 5G O-RAN Networks

Alessio Vicario*
alessio.vicario@unipa.it
University of Palermo
Palermo, Italy

Antonino Pagano
antonino.pagano@unipa.it
University of Palermo and CNIT
Palermo, Italy

Alessandra Dino
alessandra.dino01@unipa.it
University of Palermo and CNIT
Palermo, Italy

Daniele Croce
daniele.croce@unipa.it
University of Palermo and CNIT
Palermo, Italy

Ilenia Tinnirello
ilenia.tinnirello@unipa.it
University of Palermo and CNIT
Palermo, Italy

Abstract

In an increasingly densely populated network environment, a customised handover mechanism is essential to ensure service continuity, user experience quality and more efficient use of radio resources. Compared to traditional approaches, adaptive handover enables a dynamic response to actual traffic conditions, improving overall network performance. This work explores the potential of 5G mobile networks based on Open RAN architecture, focusing on the implementation of a customised handover mechanism. Using the OpenAirInterface platform and the FlexRIC framework, an xApp was developed that can monitor user throughput in real-time and dynamically activates a handover when throughput drops below a predefined threshold. The approach demonstrates the flexibility of the O-RAN architecture and the effectiveness of standardised APIs and service models in supporting intelligent network decisions. The results show how the integration of programmable functionality in the Near-RT RIC improves network performance and enables more adaptive mobility management.

CCS Concepts

• **Networks** → **Network management**; **Programmable networks**.

Keywords

5G, O-RAN, Open Air Interface, OAI, FlexRIC, Handover, xApp, Throughput, Mobility.

WiNTECH '25, Hong Kong, China

ACM ISBN 979-8-4007-1972-1/25/11
https://doi.org/10.1145/3737895.3768302

ACM Reference Format:
Alessio Vicario, Antonino Pagano, Alessandra Dino, Daniele Croce, and Ilenia Tinnirello. 2025. Towards Intelligent Mobility Management: Customized Handover in 5G O-RAN Networks. In *ACM Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization (WiNTECH '25), November 4–8, 2025, Hong Kong, China.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3737895.3768302

1 Introduction

The advent of fifth-generation (5G) networks has revolutionized the landscape of telecommunications, enabling unprecedented levels of connectivity and data transmission. With the growing demand for fast and reliable communications, the integration of heterogeneous network technologies becomes essential. The architecture of 5G networks marks a substantial evolution in wireless communication, characterized by the implementation of small cells that improve network densification, which is essential to support different quality of service (QoS) requirements [1]. Applications such as robotic telesurgery require enhanced mobile broadband (eMBB) and ultra-reliable low-latency communication (uRLLC) for effective operation [2]. The integration of heterogeneous networks (HetNets) into the 5G framework is essential to ensure seamless mobility between macro cells, small cells, and device-to-device (D2D) communication modes [1], meeting bandwidth requirements and reducing latency.

Ultra-dense networks (UDNs) within 5G architectures improve capacity but introduce complexities such as signal interference and longer handover times [3]. The degradation of communication quality during handovers, exacerbated by unreliable RSSI readings, highlights the need for innovative management strategies [4]. Line of sight (LoS) communication interruptions due to pedestrian blocks cause signal attenuation, affecting applications that require uRLLC. Addressing these challenges is critical to the successful implementation of 5G, ensuring a consistent user experience.

On the other hand, Radio Access Networks (RAN) represent the most costly and complex aspect of deploying and operating mobile communication systems [5]. RAN slice resource management is critical for applications involving flying and ground vehicles, requiring the integration of Network Slicing and Multi-access Edge Computing (MEC) for efficient resource allocation [6]. Massive MIMO networks centered on users improve the user experience through coordinated transmission from distributed units, revolutionizing the 5G architecture [7]. The challenges of associating mobile vehicles with static fog servers during handovers highlight the need for innovative architectural solutions [8]. However, in most commercial deployments, the mechanisms governing mobility management such as handover are closed, standardized implementations that are costly and time-consuming to customize or update. This rigidity limits the ability of operators to adapt handover strategies to evolving traffic patterns, specific service requirements, or experimental features. In contrast, open-source 5G platforms, combined with the Open RAN paradigm, offer a promising path toward programmable, flexible, and real-time control of handover procedures.

Leveraging these capabilities, this work demonstrates how a custom xApp, developed using the FlexRIC framework and integrated into a full OpenAirInterface 5G stack (including both core and RAN, with a USRP B210 as gNB and a commercial smartphone as UE), can invoke and control handovers based on application-defined triggers. Beyond implementing a proof-of-concept handover logic, where the decision is triggered by Key Performance Metrics (KPM) periodically reported by the gNB, the proposed work extends the official OAI codebase to support new control messages from the xApp, enabling the selection of the target cell and custom trigger parameters. The goal is to establish a foundation for future, more sophisticated solutions, that can dynamically adapt mobility management according to real-time network conditions and service requirements.

2 Background and Related Work

Handover is a fundamental mechanism to ensure service continuity in mobile networks, allowing the transfer of a User Equipment (UE) connection from one cell to another as the user moves. The node managing the serving cell instructs the UE to measure the signal quality from neighboring cells and report the results. Based on these measurements, the node may decide to initiate a handover to a neighboring cell offering better radio conditions [1]. In current networks, handovers are typically event-triggered and rely on parameters such as the Reference Signal Received Power (RSRP), Reference Signal Received Quality (RSRQ), and QoS-based criteria, as specified by the European Telecommunications Standards Institute (ETSI) and the 3rd Generation Partnership Project (3GPP) [9].

In 5G networks, despite the introduction of network slice [10], a mechanism that enables multiple logical networks with distinct service characteristics to co-exist over the same physical infrastructure, the handover process remains essential. Slicing provides service differentiation and traffic isolation (e.g., enhanced mobile broadband, low-latency communication, massive IoT), but it does not remove the need to manage physical network resources dynamically [11, 12]. As such, a gNB may initiate a handover not only due to degraded signal conditions but also for reasons like traffic load balancing or throughput optimization. This ensures efficient use of radio resources and maintains the quality of experience, even within the same slice. One of the main disadvantages of handovers triggered by these metrics is that the user's type of service is not taken into account. For example, the handover optimization requirements for broadband access that transmits high-quality video are different from those of an autonomous vehicle. For this reason, the integration of xApps is essential, as they provide enhanced functionality and flexibility, allowing network operators to customize network behavior according to specific operational requirements and rapidly deploy new services [13].

A variety of open-source frameworks exist to support xApp development and deployment in near-real-time RIC environments, each with different trade-offs in terms of flexibility, ease of use, and compatibility with OpenAirInterface (OAI). For example, *ONF SD-RAN (D-RAN)* provides a cloud-native near-real-time RIC with strong multi-vendor support [14]. Although flexible and standards-compliant, it relies heavily on containerized microservices (typically Kubernetes based), which can complicate experimental setups and limit ease of use for rapid prototyping. Integration with OAI is possible via the standard E2 interface, but no native OAI-specific RAN agent is included. The *O-RAN Software Community (O-RAN SC)* near-RT RIC offers a feature rich, fully O-RAN compliant platform [15]. It supports multiple programming languages and service models and includes example xApps such as traffic steering with handover triggering. However, the deployment is relatively heavy, often requiring Kubernetes, and integration with OAI depends on manually setting up the OAI E2 agent. *srsRAN* implements an E2 agent in its gNB and supports KPM and RC service models, making it compatible with external RICs [16]. It is lightweight and straightforward to install, but its RIC-related capabilities and xApp examples are still limited compared to dedicated RIC frameworks. Finally, *FlexRIC* is specifically designed for seamless integration with OAI, making it particularly relevant to this work [17]. It includes a native OAI-compatible E2 agent, a compact and easily deployable near-RT RIC, and supports multiple encoding formats and pluggable service

models. It also provides ready-to-use xApp examples, including handover control and APIs in C/C++/Python, enabling rapid development.

In this paper, we build on FlexRIC and OAI to implement and test a custom handover mechanism. This combination was selected because, unlike other frameworks, it allows direct integration into OAI's 5G RAN without external bridging components, minimizing setup complexity and enabling direct extensions to the OAI codebase for new control message handling. This native compatibility provides a unique advantage for prototyping and validating new mobility management strategies in an open source 5G environment.

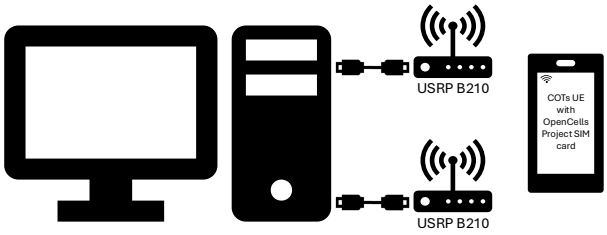


Figure 1: Setup for testing the handover procedure.

3 Methods

In cellular networks, the Radio Resource Control (RRC) protocol defines the connection states between the User Equipment (UE) and the radio access network. These states determine how the UE interacts with the network in terms of signaling, mobility, and resource allocation. The main RRC states in 5G, as defined in the 3GPP specifications [18], are `RRC_IDLE`, `RRC_CONNECTED`, and `RRC_INACTIVE`. In `RRC_IDLE`, the UE is not actively connected; in `RRC_CONNECTED`, the UE maintains an active radio connection and exchanges both control and user data with the network. The `RRC_INACTIVE` state enables fast connection resumption while reducing signaling overhead by preserving UE context at the gNB. Handover procedures are only possible when the UE is in the `RRC_CONNECTED` state [19], as the network retains full control over the radio link and can coordinate the transition of the UE between cells. In other states, such as `RRC_IDLE` or `RRC_INACTIVE`, the UE manages its mobility independently through cell reselection, and no network-initiated handover is performed.

In this work, the handover procedure was studied in the `RRC_CONNECTED` state in which the cell phone is transmitting and receiving and the network has full control of the UE connection. The procedure begins when the source node forwards a handover request, aimed at transferring control of the device, to the destination node. This message includes the destination cell identifier, the Access and Mobility Management Function (AMF) responsible for the device, as well as detailed information regarding active Protocol Data Unit (PDU) sessions and associated Quality of Service (QoS) flows.

3.1 Experimental Setup

The experimental setup, in Fig. 1, consists of one PC running the Ubuntu operating system, equipped with the OpenAir-Interface (OAI) 5G stack and the FlexRIC Near-RT RIC SDK. On the same host, we implemented a second 5G cell. Two Software Defined Radios (SDR) are connected to the PC via USB 3.0 as Radio Unit (RU) of the implemented 5G Networks. A smartphone (Google Pixel 8a) serves as the User Equipment (UE) and is configured with OpenCell SIM card [20] to connect to the network.

On the PC, the stack includes the OAI Core Network (CN), the OAI gNB, and the FlexRIC Near-RT RIC. The Near-RT RIC is preloaded with the following xApps:

- **KPM Monitor**: Prints the Key Performance Measurement (KPM) provided by the E2 node.
- **RC Monitor**: Sends aperiodic notifications for each RRC state change.
- **RC Control**: Executes a RAN control function, specifically *QoS flow mapping configuration.*

The SDR used in the setup are two Universal Software Radio Peripheral (USRP) B210 [21], by Ettus, operating in the band n78 [22].

The USRP B210 covers a wide frequency spectrum ranging from 70 MHz to 6 GHz, making it suitable for a variety of wireless applications. It supports a bandwidth of up to 56 MHz, allowing operation with signals of considerable amplitude. The device connects to the computer via a USB 3.0 interface, providing the high data transfer rates necessary to handle large amounts of real-time data. It is programmed using the USRP Hardware Driver (UHD) software, which enables the prototyping of base stations with open-source applications [23].

In our experimental setup, we employ USRP B210 due to their low transmission power, which makes them suitable for controlled indoor testing environments. This characteristic minimizes the risk of unintentional interference with commercial cellular users and allows safe experimentation without violating regulatory constraints.

3.2 Handover with Open Air Interface

The O-RAN architectural specifications define various functions that can be implemented on E2 nodes. In this work, the E2 node corresponding to the OAI gNB includes the "RAN Control" and "KPM Monitor" functions. These modules enable both REPORT and CONTROL operations, allowing the system to collect measurements from the nodes and transmit control messages accordingly. The services provided by the E2 CONTROL function are defined in specifications [23] and

Table 1: NR CGI structure

RAN Parameter ID	RAN Parameter	RAN Parameter Value Type
1	Target Primary Cell ID	STRUCTURE
2	>CHOICE *Target Cell*	STRUCTURE
3	»NR Cell	STRUCTURE
4	»>NR CGI	ELEMENT

[24], and are offered through a set of control styles. Each control style corresponds to a specific set of actions, with each action addressing a distinct RAN functionality. Every action is associated with a defined set of RAN parameters, typically specified in a mapping table. Regarding handover management, *CONTROL Style 3: Connected Mode Mobility* (CMM) is employed. This control style provides a standardized mechanism to initiate a mobility control procedure for a User Equipment (UE) operating in the RRC_CONNECTED state. Applications of this service include:

- Initiating a handover for a selected UE towards a specific target cell.
- Initiating a conditional handover for a selected UE towards a list of candidate cells.
- Initiating a handover with Dual Active Protocol Stack (DAPS) for a selected UE towards a target cell.

After receiving the control request message, the E2 Node will invoke the handover procedures as specified in [13]. The control message may include one or more of the following Information Elements (IEs), which are structured data fields used to convey specific information within messages exchanged between network entities:

- The primary target cell ID
- The list of PDU sessions to be transferred
- The list of DRBs (Data Radio Bearers) to be transferred
- The list of secondary cells to be configured

In order to develop an xApp that leverages Control Style CMM 3, it was necessary to extend its implementation in the OAI codebase repository. Specifically, modifications were made to the *ran_func_rc.c* file, which defines all the IEs associated with each control style. By default, this file only included the IEs related to the *Radio Bearer Control* style, which is used to modify *Radio Bearer* configurations [25]. For the purpose of this work, a new IE was implemented to represent the target cell identifier, known as **NR CGI** (New Radio Cell Global Identifier). It is a globally unique identifier used in 5G networks to precisely identify a specific cell within the mobile network infrastructure. This new IE is required in the control message sent by the xApp to the E2 node to trigger the handover procedure.

The hierarchical structure of the IE, which must be duly included in the *ran_func_rc.c* file, is shown in Table 1. To implement a new style within the file, it is essential to refer to the data structures defining the IEs, which are contained in the header files of the OAI repository. The OAI source file for the Key Performance Measurement (KPM) Service Model provides several parameters that can be used as input for a handover algorithm. Specifically, the parameters exposed by OAI include:

- `DRB.PdcpSduVolume`: This parameter indicates the volume of user data (SDU) carried over the PDCP layer for both downlink and uplink on a specific DRB.
- `DRB.RlcSduDelay`: Measures the delay experienced by an SDU at the RLC layer in downlink and uplink, i.e., the time taken for data packets to be transmitted from the network to the UE.
- `DRB.UEThp`: Indicates the throughput for both downlink and uplink for a specific UE over a given DRB.
- `RRU.PrbTot`: Measures the total number of Physical Resource Blocks (PRBs) used in both downlink and uplink for a specific RRU. PRBs are radio resource units allocated to the UE for data transmission.

3.3 xApp code implementation

To enable a xApp to interact with E2 nodes within the O-RAN architecture, it is necessary to use the xApp API, which provides a set of essential functions. These functions form the core interface for managing interactions between the xApp and E2 nodes, enabling the implementation of customized control and monitoring logic within the O-RAN ecosystem.

- `init_xapp_api()`: initializes the xApp API.
- `try_stop_xapp_api()`: attempts to safely stop the xApp API.
- `e2_nodes_xapp_api()`: returns an array containing the currently available E2 nodes.
- `report_sm_xapp_api()`: returns a handle that can be used to communicate with an E2 node, based on its identifier and other specific information.
- `rm_report_sm_xapp_api()`: removes a previously obtained handle, releasing the associated resources.
- `control_sm_xapp_api()`: sends a control message to an E2 node, including commands related to specific network functions.

The procedure for performing a customised handover using xApp is represented by the flowchart in Figure 2 and the pseudo code in Algorithm1. As shown in the xApp code (see Algorithm 1), the `main` function initializes the API, stores the connected E2 nodes in an array, and initializes a mutex to ensure thread synchronization. The mutex is used to prevent unsafe concurrent access to shared resources by multiple threads, thereby ensuring correct execution and avoiding race conditions.

Subsequently, two additional functions were used, namely KPM and RC. Regarding the KPM function, a subscription message to the REPORT service is generated, specifying the parameters to be monitored. The message is then sent

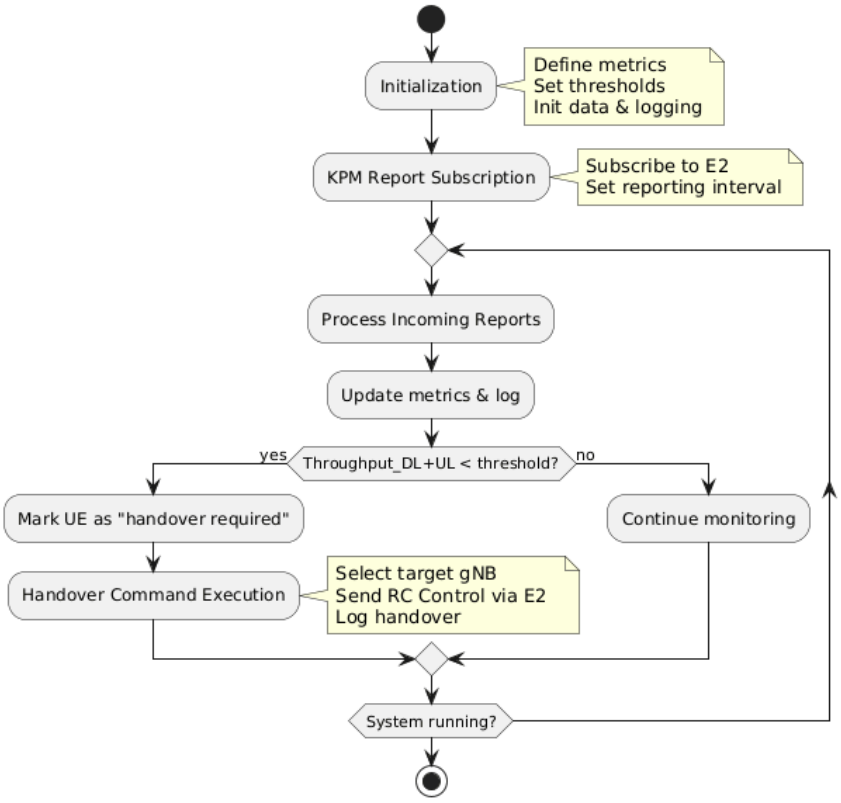


Figure 2: Flow chart for implementing the handover procedure in xApp.

using the `report_sm_xapp_api()` function, and finally, the allocated memory is properly released.

The KPM function implements the logic to send **KPM SUBSCRIPTION** messages to each connected E2 node from within an xApp. Specifically, the KPM RAN function is defined with a constant identifier 2. For every node in the `nodes` list, the structure of the node is accessed, and the index of the KPM function supported by that node is retrieved. It is then verified, using an assertion, that the identified function is indeed of type **KPM**. If the node provides at least one reporting style (i.e., the `ric_report_style_list` is not null), a KPM subscription message (`kpm_sub`) is generated and sent to the node using the `report_sm_xapp_api`. The success of the operation is asserted, and finally, the memory allocated for the subscription message is released. This procedure dynamically enables the monitoring of performance metrics across multiple network nodes using the KPM (Key Performance Measurement) service within the O-RAN architecture.

RC code block begins with the creation of a subscription to the RC reporting service. This service enables the reception of asynchronous reports whenever the RRC state of a selected UE changes—for example, transitioning from `CONNECTED` to `IDLE`.

RC RAN code block defines a constant for the RC RAN function with value 3 and iterates on all connected E2 nodes. For each node, it locates the index of the RC function among the supported RAN functions. An assertion verifies that the identified function is indeed RC. If the node supports an RC reporting service (indicated by a non-null report), a subscription message is generated for this service. This message is sent to the node via a dedicated API, and the success of the operation is checked. Finally, the memory allocated for the subscription message is freed, ensuring proper resource management.

4 Results

We exploited the parameters listed in subsection 3.2 to design a customised handover algorithm. Specifically, the KPM parameters were measured by simulating various traffic and UE mobility conditions within the cell. The goal was to analyse the behaviour of these parameters and identify any significant variations that could indicate useful triggers for the handover decision-making process. From a series of preliminary experiments, it was observed that the distance of the UE from the gNB resulted in some parameter variations that

Algorithm 1: Pseudocode for Managing KPM and RC Subscriptions in an xApp

```
Input: Command-line arguments
Result: KPM and RC subscriptions are created and
        sent for each connected E2 node
function main(argc, argv):
    args ← init_fr_args(argc, argv) ;
    init_xapp_api(args) ;
    /* Initialize synchronization structures
    */ Initialize mutex with default attributes ;
    Ensure mutex initialization succeeded ;
    hndl ← allocate array of size nodes.len ;
    Wait for 1 second ;
    nodes ← e2_nodes_xapp_api() ;
      /* Handle KPM RAN function (ID = 2) */
    for i ← 0 to nodes.len − 1 do
        n ← nodes.n[i] ;
        idx ← find_sm_idx(n.rf, n.len_rf, eq_sm, 2) ;
        if n.rf[idx].defn.type == KPM then
            if n.rf[idx].defn.kpm.ric_report_style_list ≠
              NULL then
                kpm_sub ←
                  gen_kpm_subs(n.rf[idx].defn.kpm) ;
                hndl[i] ← report_sm_xapp_api(n.id,
                  2, kpm_sub, sm_cb_kpm) ;
                Ensure hndl[i] is successful ;
                Free kpm_sub data ;
    /* Handle RC RAN function (ID = 3) */ for
    i ← 0 to nodes.len − 1 do
        n ← nodes.n[i] ;
        idx ← find_sm_idx(n.rf, n.len_rf, eq_sm, 3) ;
        if n.rf[idx].defn.type == RC then
            if n.rf[idx].defn.rc.report ≠ NULL then
                rc_sub ←
                  gen_rc_sub_msg(n.rf[idx].defn.rc.report)
                  ;
                hndl[i] ← report_sm_xapp_api(n.id,
                  3, rc_sub, sm_cb_rc) ;
                Ensure hndl[i] is successful ;
                Free rc_sub data ;
    Ensure allocation succeeded ;
```

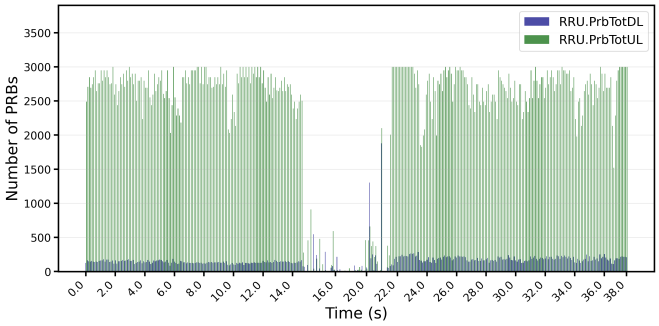
can be used to trigger a handover. Although several (or all) parameters analysed could be used, to demonstrate the possibility of performing a customised handover using xApp, the most responsive parameter, throughput, was selected. We used the iPerf software to operate in a client-server model, where one device acts as a server, configured to listen and receive data, in this case the core network (CN), while the other device, the UE user equipment, acts as a client, sending data to the server in order to measure the data transfer rate.

The status transition message from Connected to Idle was used to terminate the actual speed monitoring cycle. This cycle triggers the sending of the RC CONTROL message, containing the handover control, only when the actual downlink or uplink speed exceeds a predefined threshold. The threshold value was set to 1000 kbps. Once the control message is sent, the cycle is interrupted, memory is freed, and the xApp is terminated using the corresponding API function.

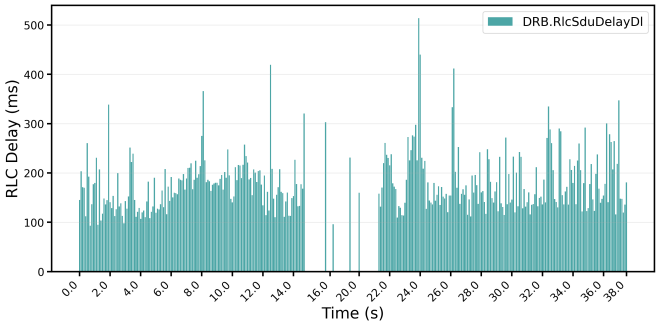
Figure 3 shows the variation of the parameters monitored during a handover. In particular, the throughput graph, depicted in Figure 3c, clearly shows three distinct phases: (i) An initial phase in which the throughput remains relatively stable around 6000–8000 Kbps, with some fluctuations typical of variable radio conditions. (ii) The handover phase, during which the throughput drops below 1000 Kbps; in this case, a brief interruption of throughput is observed, dropping practically to zero. This is the critical moment when the cell change occurs. (iii) The post-handover phase, in which the throughput quickly recovers and reaches significantly higher values (12000–15000 Kbps), indicating that the device has connected to a cell with better radio conditions. Similarly, in Figure 3d, the PDCP Volume graph exhibits a comparable pattern. In the pre-handover phase, the volume remains constant but moderate, around 4–6 bytes. During the handover, a significant drop in the transmitted data volume is observed, reflecting the temporary disruption in the connection. In the post-handover phase, the volume rises sharply to 10–12 bytes, in direct correlation with the improvement in throughput, indicating enhanced radio conditions in the new serving cell.

Fig.4 shows a captured sequence of NGAP (Next Generation Application Protocol) messages exchanged between different network nodes during the initiation of a handover procedure. The process begins with a `HandoverRequired` message sent from the source gNB (`192.168.70.141`) to the AMF (`192.168.70.132`), indicating that the UE needs to be handed over to a target gNB. The AMF forwards this request to the target gNB (`192.168.70.140`) via a `HandoverRequest`. After the target gNB accepts the request, it responds with a `HandoverCommand` (preceded by a TCP Selective Acknowledgement). Once the UE has successfully accessed the target cell, the target gNB sends a `HandoverNotify` message to the AMF. The process concludes with a `PathSwitchRequest` and an `UplinkNASTransport` message, finalizing the switch of the user plane path to the new gNB.

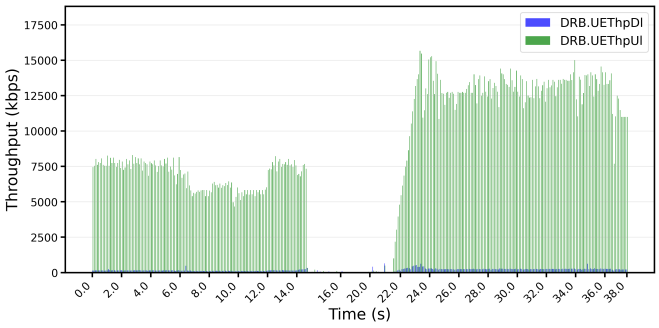
Overall, the results indicate that the customized handover process was successfully executed. The transition between



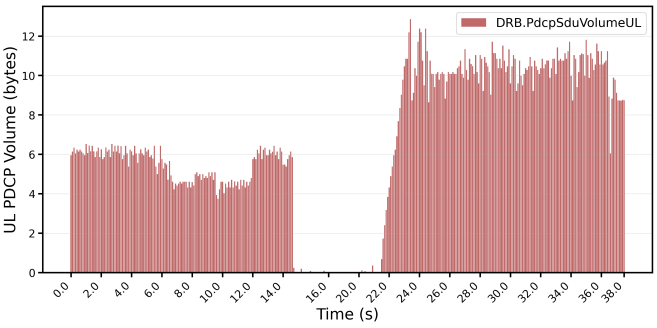
(a) The total number of Physical Resource Blocks (PRBs) used in both downlink and uplink for a specific RRU.



(b) Delay experienced by an (SDU) at the RLC layer in downlink.

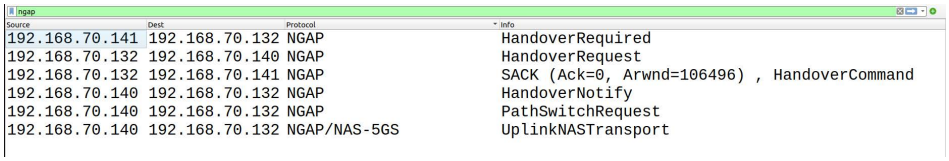


(c) Throughput (downlink and uplink) for a specific UE over a given DRB, measured using DRB.UEThp.



(d) Volume of user data (SDU) transmitted over the PDCP layer in the uplink for a specific DRB.

Figure 3: KPM parameters analysed as input for the handover procedure.



Source	Dest	Protocol	Info
192.168.70.141	192.168.70.132	NGAP	HandoverRequired
192.168.70.132	192.168.70.140	NGAP	HandoverRequest
192.168.70.132	192.168.70.141	NGAP	SACK (Ack=0, Arwnd=106496) , HandoverCommand
192.168.70.140	192.168.70.132	NGAP	HandoverNotify
192.168.70.140	192.168.70.132	NGAP	PathSwitchRequest
192.168.70.140	192.168.70.132	NGAP/NAS-5GS	UplinkNASTransport

Figure 4: Wireshark message captured during handover procedure.

cells caused only minimal service interruption, and the post-handover phase was characterized by a marked improvement in performance. This improvement is supported by increases in both transmission speed and PDCP volume, confirming the reliability of the measurements. Furthermore, the results suggest that implementing a customized handover strategy via an xAPP can effectively guide the device to a 5G cell that offers better coverage or reduced congestion. Such optimization maximizes network resource utilization and can significantly improves the end-user. More complex optimizations are also possible, which is left for future work.

5 Conclusion

This study investigates the potential of 5G Open RAN networks through the implementation of a customized handover algorithm, developed using the OpenAirInterface (OAI) platform and the FlexRIC SDK. The proposed algorithm leverages throughput as the primary decision-making parameter for handover execution. The obtained results clearly demonstrate the flexibility and adaptability of open-source architectures such as OAI, highlighting the ease of integrating new xApps for more intelligent and dynamic network management. This capability paves the way for optimizing network performance based on customized criteria and the evolving needs of users and services. This work represents a first step toward exploring the capabilities of 5G Open RAN networks.

Future research may focus on implementing more sophisticated handover algorithms that combine multiple decision parameters and leverage machine learning techniques to dynamically adapt to network conditions. Additional developments could also explore the integration of further xApps to enable advanced functionalities such as network slicing, mobility control, and radio resource optimization.

Acknowledgments

This work was supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU: partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART"), S2 SUPER – Programmable Networks, Cascade project PRISM - CUP: C79J24000190004 and S14 Net4Future, Cross-project vision and results - CUP: D93C22000910001; and project PNRR M4 - C2 - investment 1.1: Projects of Significant National Interest (PRIN) - PRIN 2022 PNRR code P2022WA578 "BISS: Beyond-5G Infrastructure for Spectrum Sensing", CUP B53D23024110001 and PRIN 2022 code 2022FYCNPT "IoT-SensE: IoT-based Sensing Extension", CUP B53D23002610006.

References

[1] Amiraslan Haghrah, Mehran Pourmohammad Abdollahi, Hosein Azarhava, and Javad Musevi Niya. 2023. A survey on the handover management in 5g-nr cellular networks: aspects, approaches and challenges. *EURASIP Journal on Wireless Communications and Networking*, 2023, 1, 52.

[2] Qi Zhang, Jianhui Liu, and Guodong Zhao. 2018. Towards 5g enabled tactile robotic telesurgery. *arXiv preprint arXiv:1803.03586*.

[3] Donglin Wang, Anjie Qiu, Qiuheng Zhou, Sanket Partani, and Hans D Schotten. 2023. Investigating the impact of variables on handover performance in 5g ultra-dense networks. In *2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. IEEE, 567–572.

[4] Sunil Mathew, Eleonora Oliosi, Luca Davoli, Nicolò Strozzi, Andrea Notari, and Gianluigi Ferrari. 2024. Csi-rsrp-based unnecessary handover mitigation through linear regression in dynamic 5g nr environments. *IEEE Access*.

[5] Aleksandra Checko, Henrik L Christiansen, Ying Yan, Lara Scolari, Georgios Kardaras, Michael S Berger, and Lars Dittmann. 2014. Cloud ran for mobile networks—a technology overview. *IEEE Communications surveys & tutorials*, 17, 1, 405–426.

[6] Xuming Yao, Yingyng Sun, and Tao Han. 2018. Study on energy consumption and coverage of hierarchical cooperative of small cell base stations in heterogeneous networks. In *2018 20th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 513–518.

[7] Hussein A Ammar, Raviraj Adve, Shahram Shahbazpanahi, Gary Boudreau, and Kothapalli Venkata Srinivas. 2021. User-centric cell-free massive mimo networks: a survey of opportunities, challenges and solutions. *IEEE Communications Surveys & Tutorials*, 24, 1, 611–652.

[8] Salman Memon and Muthucumaru Maheswaran. 2019. Using machine learning for handover optimization in vehicular fog computing. In *Proceedings of the 34th ACM/SIGAPP symposium on applied computing*, 182–190.

[9] Mikel Irazabal. 2023. *OAI Webinar Series – Chapter Five: FlexRIC Tutorial – xApp Development*. OpenAirInterface Webinar Series. (Feb. 2023).

[10] 3rd Generation Partnership Project (3GPP). 2024. System Architecture for the 5G System (5GS). Tech. rep. TS 23.501. Version 16.6.0. 3GPP.

[11] Domenico Garlisi, Antonino Pagano, Fabrizio Giuliano, Daniele Croce, and Ilenia Tinnirello. 2025. Interference analysis of lorawan and sigfox in large-scale urban iot networks. *IEEE Access*, 13, 44836–44848.

[12] Farzam Nosrati, Emebet Gelaw, Roberto Corallo, Silvia Schilleci, Alessio Vicario, and Daniele Croce. 2024. Cooperative spectrum sensing for beyond-5g networks in fading environments. In *Proceedings of the 25th International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing* (MobiHoc '24). ACM, Athens, Greece, 446–451.

[13] Joao F Santos, Alexandre Huff, Daniel Campos, Kleber V Cardoso, Cristiano B Both, and Luiz A DaSilva. 2025. Managing o-ran networks: xapp development from zero to hero. *IEEE Communications Surveys & Tutorials*.

[14] Open Networking Foundation. 2023. *ONF SD-RAN Project*. Accessed: 2025-08-14. https://opennetworking.org/news-and-events/press-releases/onfs-sd-ran-now-fully-released-to-open-source/#:~:text=ONF%E2%80%99s%20SD,consortium%20of%20network%20operators%20who.

[15] O-RAN Software Community. 2025. *O-RAN Software Community Near-RT RIC*. Accessed: 2025-08-14. https://docs.srsran.com/projects/project/en/latest/tutorials/source/near-rt-ric/source/index.html.

[16] Software Radio Systems. 2025. *srsRAN Project*. Accessed: 2025-08-14. https://docs.srsran.com/projects/project/en/latest/tutorials/source/near-rt-ric/source/index.html.

[17] EURECOM. 2025. *FlexRIC - EURECOM's Open-Source O-RAN RIC*. Accessed: 2025-08-14. https://openairinterface.org/news/oai-integrates-o-ran-ric-with-its-5g-stack-and-showcases-monitoring-control-xapps/#:~:text=The%20RT%20controller%20component%20of,replace%20the%20FlexRIC%20RT%20controller.

[18] 3GPP. 2024. NR; Radio Resource Control (RRC); Protocol specification. Tech. rep. TS 38.331. Version 17.1.0. 3GPP.

[19] Srinath Potnuru and Prajwol Kumar Nakarmi. 2021. Berserker: asn. 1-based fuzzing of radio resource control protocol for 4g and 5g. In *2021 17th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 295–300.

[20] OpenCell Technologies. 2024. *OpenCell Project*. Accessed: 2025-08-02.

[21] Ettus Research. [n. d.] *USRP™ B200/B210 Bus Series*. Ettus Research. https://www.ettus.com/wp-content/uploads/2019/01/b200-b210%5C_spec%5C_sheet.pdf.

[22] 3GPP. 2025. User Equipment (UE) radio transmission and reception; Part 1: Range 1 Standalone. Technical Specification (TS) 38.300. 18.2.0. 3rd Generation Partnership Project (3GPP), (July 2025).

[23] Ettus Research. 2025. *USRP N310 – Ettus Research*. Available online: https://www.ettus.com/all-products/usrp-n310/ [Accessed: 23-May-2025].

[24] Open Cells Project. 2025. *SIM cards – 4G and 5G reference software*. Accessed: 2025-07-08.

[25] Michele Polese, Leonardo Bonati, Salvatore D'oro, Stefano Basagni, and Tommaso Melodia. 2023. Understanding o-ran: architecture, interfaces, algorithms, security, and research challenges. *IEEE Communications Surveys & Tutorials*, 25, 2, 1376–1411.