# HawkVision: Network Coding in O-RAN

Osel Lhamo*, Omer H. Khan*, Tung V. Doan†, Elif Tasdemir*, Giang T. Nguyen†‡, Frank H.P. Fitzek*‡
*Deutsche Telekom Chair of Communication Networks, Technische Universität Dresden, 01062, Dresden, Germany
†Haptic Communication Systems, Technische Universität Dresden, 01062, Dresden, Germany
‡Centre for Tactile Internet with Human-in-the-Loop (CeTI)
E-mails: {osel.lhamo|omer_hanif.khan|tung.doan_van|elif.tasdemir|giang.nguyen|frank.fitzek}@tu-dresden.de

***Abstract*—Recent advances in networking technologies, such as in-network computing (INC), have demonstrated significant potential for mobile networks. Among these, Random Linear Network Coding (RLNC), a class of forward error correction codes, has proven especially promising for enhancing reliability. RLNC reduces latency by transmitting additional coded packets, making it well-suited for supporting Ultra-Reliable Low Latency Communications (URLLC), a key requirement in next-generation mobile networks. To fully harness the benefits of RLNC in mobile networks, it is essential to closely monitor its operations to design effective RLNC schemes. This need arises from key factors, including the dynamic nature of mobile environments and the continuous emergence of new applications. However, achieving this in traditional mobile networks remains challenging due to their closed architectures and vendor lock-in. We propose *HawkVision*, a monitoring solution for RLNC operations that leverages the flexibility of Open Radio Access Network (O-RAN). *HawkVision* supports monitoring various RLNC schemes, such as Sliding Window and Systematic Block Code, within mobile networks. It uses xApps in O-RAN to observe RLNC behavior in the RAN. We implement *HawkVision* using FlexRIC, a widely used O-RAN platform, and deploy a Key Performance Metric (KPM) xApp to collect relevant metrics. Testbed results demonstrate *HawkVision*'s effectiveness in monitoring the operations for different RLNC schemes.**

***Index Terms*—URLLC, Random Linear Network Coding, Beyond 5G, INC, RIC, O-RAN, OpenAirInterface**

## I. Introduction

Mobile networks have rapidly evolved to meet the ultra-reliable low-latency communication (URLLC) demands of an ever-growing range of emerging applications, such as the metaverse. This evolution paves the way for a wide range of new technological advancements, particularly in-network computing (INC), which plays an important role in integrating data processing directly within the network infrastructure [1, 2]. Within this paradigm, Network Coding (NC), particularly Random Linear Network Coding (RLNC), as deployed in the network, has gained attention for its potential to improve reliability and reduce latency [3]. RLNC is a forward error correction mechanism [4] that combines original packets (from the sender) with random coding coefficients to produce coded redundant packets. The receiver uses received coded redundant packets and original packets to decode any corrupted or lost original packets without extra signalling or coordination between the sender and the receiver. Multiple RLNC variations exist, among which Sliding Window and Systematic Block Code are particularly known to offer high reliability while performing low packet delay [5, 6], contributing to the fulfillment of URLLC requirements.

While the integration of RLNC into mobile networks brings promising improvements, it also requires additional physical resources, particularly in the Radio Access Network (RAN), such as increased bandwidth usage due to the overhead of transmitting coded packets. This overhead may appear negligible in isolated instances, but it can impose substantial penalties on spectrum utilization and overall network throughput, particularly in dense user deployments or bandwidth-constrained scenarios. Monitoring the operations of RLNC can help identify such scenarios and mitigate their impact in real time. Without continuous monitoring, network operators can risk over-provisioning redundancy, or applying RLNC when it is not needed. Consequently, monitoring RLNC is essential for improving resource efficiency, a critical challenge in mobile networks [7, 8]. Moreover, monitoring ensures the proper behavior of its operations and enables adaptation to diverse emerging applications.

A traditional mobile network consists of an inherently closed and proprietary architecture with limited interoperability and restricted access to internal components. This lack of visibility and control hinders the integration of custom monitoring solutions, especially since these systems often rely on vendor-specific tools and interfaces. To address these shortcomings, we propose *HawkVision*, an efficient monitoring solution designed to observe RLNC operations within mobile networks by leveraging Open Radio Access Network (O-RAN). This architecture is vital to achieve an increasingly open, resilient, sustainable, and intelligent mobile network [9]. It allows the seamless integration of third-party tools by supporting the disaggregation of network services and granting access to internal control and data points. This openness is important for observing the behavior and performance of RLNC under realistic mobile network conditions, facilitating fine-grained analysis and optimization. Specifically, *HawkVision* consists of NC xApps in the Near-Real-Time RIC (Near-RT RIC) and RLNC components in the user plane of the mobile network. We implement *HawkVision* using FlexRIC [10] framework with Key Performance Metric (KPM) xApp to enable near-real-time monitoring of RLNC operations. The results we obtained from a practical testbed demonstrated *HawkVision*'s ability to observe performance of RLNC across various metrics in the RAN.

978-3-903176-75-1 ©2025 IFIP

## II. Related Work

RLNC has emerged as a key technique for addressing reliability challenges in mobile networks. For instance, Mao et al. [11] used linear network coding to enhance reliability and latency in integrated access and backhaul networks. Shahzad et al. [3] proposed a learning-based RLNC framework to reduce latency and decoding complexity for URLLC scenarios, such as the tactile internet. Peng et al. [12] proposed a physical layer NC scheme for Network MIMO that reduces backhaul load using optimized binary mappings. Vukobratović et al. [13] applied RLNC to mobile video delivery, demonstrating improved reliability and multicast efficiency in 5G System (5GS). However, these approaches rely on simulations and require significant changes to the RAN protocol stack, thus limiting real world applicability.

We recently proposed adding RLNC NC in a cloud-native mobile networks as a plug-and-play feature that requires minimal changes to the current architecture [14]. The encoder runs in a Docker container before the User Plane Function (UPF), giving operators flexibility to deploy or scale as needed. The decoder is installed on the User Equipment (UE) like any other app, allowing it to decode and recover lost packets easily. Focusing on the downlink communication, various senders in the external Data Network (ext-DN), such as the Internet, send user packets to UEs. This user packets first goes through the encoder, which adds coded redundant packets to the original data before sending it to the UPF. The UPF treats both types of packets the same, adding a tunneling header and sending them to the gNodeB (gNB). The gNB removes the header and forwards the packets to the UE over the air. The decoder on the UE then uses the redundant packets to recover any lost or damaged data. Although this work demonstrates the smooth integration of RLNC in mobile networks, it lacks the means to monitor its operations.

## III. Background

The O-RAN supports a disaggregated architecture [15], separating hardware from software and allowing multi-vendor integration. At the core of O-RAN's architecture is the RAN Intelligent Controller (RIC), which enables more flexible and automated control of the RAN, facilitating network operators to optimize performance and adapt to changing conditions. The RIC supports mobile broadband, network slicing, mission-critical communications, and other features compatible with 3GPP Release 15 and beyond[1]. There are two types of RIC: Non-Real-Time RIC (Non-RT RIC) and Near-RT RIC. The Non-RT RIC operates on timescales longer than one second. Meanwhile, the operation time of the Near-RT RIC ranges from 10 milliseconds to one second, enabling near-real-time control and optimization of RAN features. As shown in Fig. 1, communication between the Non-RT RIC and the Near-RT RIC occurs through the A1 interface.

The Near-RT RIC can address network demands in near-real-time as it hosts specialized software applications, known as xApps, which are modular programs designed to enhance network performance by enabling key functions such as mobility management, radio resource management, and security. xApps interact with the underlying RAN infrastructure through open interfaces such as the E2 interface, facilitating seamless communication and control. The Near-RT RIC also includes key components, such as the messaging infrastructure that supports interaction between different internal functions, as well as the database and the shared data layer to collect RAN or UE data. With these components, O-RAN can support various use cases, such as quality of service (QoS) resource allocation, energy efficiency, and traffic steering [9, 15], all of which depend on effective monitoring. Consequently, recent research prioritizes monitoring, as it not only enables visibility into RAN metrics but also underpins intelligent control and decision-making within the RAN [16, 17].

[1]https://www.viavisolutions.com/en-us/ran-intelligent-controller

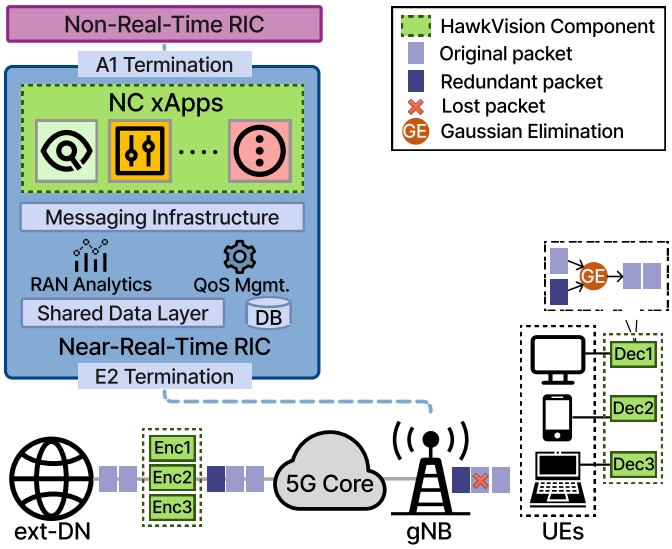


Fig. 1. The architecture design of *HawkVision* illustrating near-real-time monitoring of resource consumption associated with RLNC operations in a 5G system using the NC xApps within the O-RAN framework. Packets are transmitted from the ext-DN which passes through encoder. The generated coded packets with the original packets are sent via the 5G Core and the gNB to different UEs where decoding occurs.

## IV. HawkVision: Design and Implementation

We propose *HawkVision* that provides a monitoring solution for the RLNC operations in mobile networks, leveraging the flexibility of O-RAN. We start by presenting system design of *HawkVision*, providing the key design challenges, followed by a detailed description of its implementation.

### A. Design

For the effective monitoring of RLNC operations in mobile networks, our design must address two key requirements. Firstly, *HawkVision* must support the seamless integration of RLNC elements in mobile networks without disrupting the existing architecture, allowing backward compatibility and low operational overhead. Secondly, the solution must enable the near-real-time collection of relevant RAN metrics that reflects

how RNLC operations affect the handling and responsiveness of mobile networks. This includes capturing variations in traffic volume, resource utilization, and transmission performance, which give insights into RNLC behaviors under real-world conditions and facilitate continuous optimization.

To address the first requirement, we have to identify where to deploy RLNC, which is not a monolithic entity, but consists of at least an encoder and a decoder. One deployment strategy might involve placing the encoder in the gNB and the decoder in the UEs, given the reliability concerns due to the wireless communication between them. However, this deployment is complicated by tunneling mechanisms which encapsulate user data and limit visibility into the packet content. Therefore, we adopt our previous approach briefly mentioned in Section III by integrating the encoder between the UPF and the ext-DN, where traffic is more easily accessible for RLNC processing. The decoder is installed as an application in the UE, allowing end-to-end RLNC operation in the mobile network.

The challenge of addressing the second requirement is where we can collect near-real-time information on RLNC operations. Deploying additional components for monitoring will be challenging, potentially requiring modifications to the existing mobile network architecture. Even with the seamless integration of additional monitoring components for RLNC operations, we still lack interfaces to access mobile network metrics due to restrictions imposed by network operators in traditional mobile networks. O-RAN helps resolve this issue by introducing open and standardized interfaces and disaggregated components, facilitating greater visibility and control. Furthermore, the Near-RT RIC supports the collection of near-real-time information, which helps overcome the limitations of monitoring in conventional operator-managed mobile networks.

Building on this, we design the architecture of *HawkVision* as shown in Fig. 1. It consists of NC xApps that operate within the Near-RT RIC, along with RLNC components integrated into the user plane of the mobile network. We chose to develop NC xApps as it seamlessly integrates with existing network architectures, eliminating the need for modifications. Specifically, NC xApps collect RAN performance metrics from gNB through the E2 interface, where predefined Service Models describe messages exchanged between the gNB and the Near-RT RIC. The gNB can periodically send the required RAN performance metrics such as signal quality, radio resource utilization, mobility data, traffic load, latency metrics, connection statistics, failure indicators, and interference levels to the NC xApps, which can use these metrics to provide near-real-time information about RLNC operations.

### B. Implementation

The implementation of *HawkVision* requires both O-RAN and RLNC. For our O-RAN system, we use OpenAirInterface (OAI) [18] to implement 5G components with O-RAN compliant interfaces and integrate them with Flexible RIC (FlexRIC) [10] to realize the Near-RT RIC platform. We choose FlexRIC because it is modular, lightweight, resource-efficient, and fast compared to other popular alternatives (for example, OSC-RIC[2] and $\mu$ONOS-RIC[3]). However, due to the design of *HawkVision*, it can be easily integrated into other implementations of 5G components and Near-RT RIC. FlexRIC offers xApps in C/C++ and Python and implements different Service Models[4]. In particular, we use KPM xApp to monitor RLNC operations because it is based on the standardized E2 Service Model, defined by the O-RAN Alliance [19].

For the implementation of RLNC protocols, specifically Sliding Window (SW) and Systematic Block Code (SBC), we employ a C++ library with a Python wrapper. The difference between SW and SBC is in how they handle packet encoding and transmission. SW dynamically encodes packets within a variable-sized window that advances as transmission progresses, allowing real-time adaptation to network conditions, without fixed block boundaries. Thus, it is well suited for latency-sensitive applications like live video streaming [20]. Meanwhile, SBC divides the data into fixed-size generations, where each encoded packet is generated from its assigned block. It first transmits the original packets, followed by coded combinations, which introduce latency due to the wait for complete block assembly. It is more effective for fixed-size data transmissions, such as pre-recorded video streaming [3], where full block reception enables efficient decoding.

## V. Performance Evaluation

In this section, we assess the performance of RLNC using *HawkVision*. Specifically, our aim is to answer the following questions: (i) what are the radio resource costs associated with the integration of RLNC in mobile networks? and (ii) how different RLNC schemes affect various performance metrics of mobile networks when configured with various parameters under different traffic types?

### A. Testbed

In the following sections, we discuss the essential components, including hardware, software, and deployment strategies, necessary to build and operate our testbed.

*1) Hardware:* Fig. 2 shows our testbed that comprises of three general-purpose hosts connected by Aruba 2930F[5] switch to emulate configurations commonly found in real-world environments. Host 1 and 2 have Intel Core i7-6700 CPUs, 32GB RAM, and run Ubuntu 20.04.6. Host 3 has an Intel Core i5-7260U CPU, 4GB RAM, and run Ubuntu 22.04.3; used primarily for core deployment because it has lower performance requirements. Each host has two network interfaces: one for management network and one for supporting control and user plane traffic.

*2) Software:* We deploy core and RAN in OAI using Docker containers, with Docker Compose used to orchestrate and streamline the deployment. The OAI framework's simulation tool [21] provides the noise model, which from our

[2]https://docs.o-ran-sc.org/en/latest/projects.html
[3]https://docs.onosproject.org/v0.6.0/onos-cli/docs/cli/onos_ric/
[4]https://gitlab.eurecom.fr/mosaic5g/flexric
[5]https://www.arubanetworks.com/resource/2930f-switch-series-data-sheet/

observations, lacks stability. Thus, we use tc-netem[6] as an alternative. This approach allows us to implement rules for randomly dropping packets in the UPF container to analyze RLNC under realistic packet loss conditions. Specifically, we configured tc-netem to simulate a 10% random packet loss, a rate commonly considered in previous research [4, 5, 22]. We use tcpreplay[7], an open source tool, to generate traffic by playing back packet capture files.

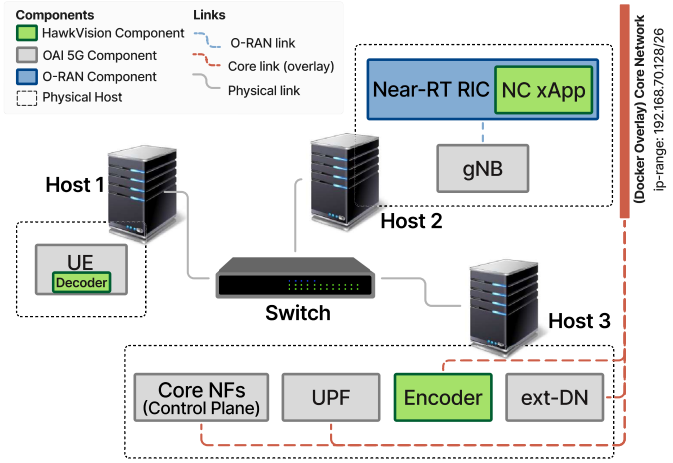


Fig. 2. Testbed consisting of three hosts housing Core NFs (Control Plane), UPF, ext-DN, gNB, UE, Near-RT RIC (with xApp), Encoder, and Decoder that are connected via a switch at the center.

*3) System deployment:* As shown in Fig. 2, the Core Network Functions (NFs), including both the control plane and the user plane, are deployed on Host 3 as Docker containers. Meanwhile, the UE and the gNB are run on bare-metal on Host 1 and 2, respectively, using the OAI deployment[8]. We integrated the encoder container between the ext-DN and the UPF container. The decoder library is installed in Host 1 and used by the UE for decoding the received packets. To maintain a controlled testing environment and avoid influence from external factors affecting radio resource usage, only one UE was used in the setup. This allows for a focused evaluation of network coding performance.

## B. Performance metrics

Previously, we demonstrated the efficiency of RLNC in enhancing reliability and reducing latency in mobile networks [14]. In this evaluation, we evaluated various performance metrics to assess the operation of RLNC using *HawkVision*. We focus on the downlink communication because the majority of user data traffic flows in the downlink. Therefore, the monitored metrics captured at gNB are specific to the downlink as shown below:

[6]https://man7.org/linux/man-pages/man8/tc-netem.8.html
[7]https://tcpreplay.appneta.com
[8]https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/openair2/

*a) PDCP SDU Volume:* The Packet Data Convergence Protocol (PDCP) layer provides crucial metrics related to data transmission [23], including header compression efficiency, encryption performance, and retransmission statistics [24]. Monitoring PDCP Service Data Unit (SDU) volume offers valuable insight into upper-layer performance. It is calculated in KPM xApp [25] using $\text{pdcp}_{sdu\_vol} = \frac{(\text{rxpdcp}(t) - \text{rxpdcp}(t-\Delta t)) \times 8}{1000}$, where $\text{rxpdcp}(t)$ denotes the total number of bytes received by the PDCP layer at the current time, and $\text{rxpdcp}(t-\Delta t)$ represents the stored value from the previous measurement period.

*b) Total PRB Utilization:* Physical Resource Block (PRB) is the smallest resource allocation unit in RAN[9]. Evaluating PRB utilization helps to assess how efficiently radio resources are allocated and whether the system can maintain the required QoS under varying traffic demands [24]. This is calculated in KPM xApp [25] as: $\text{prb}_{use} = \text{total\_prb}_{DL}(t) - \text{total\_prb}_{DL}(t-\Delta t)$, where $\text{total\_prb}_{DL}(t)$ refers to the total number of PRBs in the downlink at the current time, and $\text{total\_prb}_{DL}(t-\Delta t)$ is the corresponding stored value from the previous measurement period.

*c) UE Throughput:* UE throughput measures the effective data transfer rate to end-user devices, providing insights into user experience and the efficiency of data delivery across the network [24]. For the downlink transmission, it is calculated in KPM xApp as described in [25] using: $\text{ue}_{thp} = \frac{\text{txrlc}(t) - \text{txrlc}(t-\Delta t) \times 8}{\Delta t}$, where $\text{txrlc}(t)$ represents the cumulative number of bytes transmitted by the radio link control layer at the current time, and $\text{txrlc}(t-\Delta t)$ is the stored value from the previous measurement period.

## C. Traffic types

To simulate realistic conditions, we used a variety of traffic types to replicate real-world scenarios and provide meaningful experimental results. We provide the characteristics of the different traffic types in Table I.

| Traffic | Payload Size (B) | Inter-Packet Delay (ms) | Total Pkts. |
|---|---|---|---|
| Audio | 480 | 1 (constant) | 26425 |
| Haptic | 82 | 24 (constant) | 635000 |
| Video | 1328 (max) | [0.001, 210) (variable) | 11942 |

TABLE I
THREE TRAFFIC TYPES USED IN THE EVALUATION.

## D. RLNC configuration

As mentioned in Section IV-B, we monitor two RLNC schemes: Sliding Window (SW) and Systematic Block Code (SBC). In both the schemes, it is important to ensure that the payload size remain consistent between the encoder and the decoder. Both schemes have payload sizes tailored to the traffic type, set at 1328 B for video, 480 B for audio, and 82 B for haptic traffic [14]. As shown in Table II, we examine seven scenarios by changing the configuration parameters of RLNC schemes: i) one baseline scenario (*No RLNC*),

[9]https://www.gaussianwaves.com/2022/02/5g-nr-resource-block/

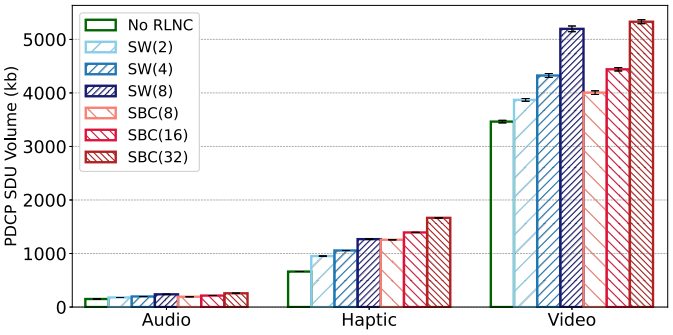


Fig. 3. Mean PDCP SDU volume monitored for various traffic

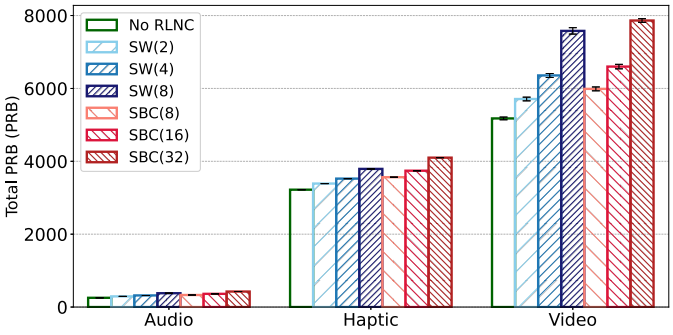


Fig. 4. Mean Total PRB usage monitored for various traffic

ii) three scenarios with SBC (*SBC(8)*, *SBC(16)*, and *SBC(32)*, with generation size = 64 and redundancy = 8, 16, and 32, respectively), and iii) three scenarios with SW (*SW(2)*, *SW(4)*, and *SW(8)*, with batch size = window size = 16 and redundancy = 2, 4, and 8, respectively, chosen to provide a comparable amount of redundancy to the SBC scenarios).

| **Scenario** | **RLNC Parameter** | | |
|---|---|---|---|
| No RLNC | *None (0% Redundancy)* | | |
| | **Original Pkts** | **Coded Pkts** *(Redundancy %)* | |
| SBC(8) | 64 | 8 *(12.5 %)* | |
| SBC(16) | 64 | 16 *(25 %)* | |
| SBC(32) | 64 | 32 *(50 %)* | |
| | **Original Pkts** | **Coded Pkts** *(Redundancy %)* | **Window** |
| SW(2) | 16 | 2 *(12.5 %)* | 16 |
| SW(4) | 16 | 4 *(25 %)* | 16 |
| SW(8) | 16 | 8 *(50 %)* | 16 |

TABLE II
MEASUREMENT SCENARIOS AND REDUNDANCY LEVELS.

### *E. Evaluation process*

With the focus on monitoring RLNC operations, we disable retransmissions at both the radio link control and the medium access control layers by configuring gNB to operate in the unacknowledged mode [26] and limiting hybrid automatic repeat request rounds to one. We generate traffic in ext-DN using one of the pre-selected packet capture files via tcpreplay and send it to the UE. During transmission, the traffic is subjected to random packet loss at the UPF using tc-netem. In the baseline scenario, the traffic is sent from the ext-DN to the UE without experiencing RLNC-related operations. For the subsequent remaining scenarios, the traffic from the ext-DN undergoes RLNC encoding and is forwarded through the UPF and gNB. Upon arrival at the UE, the decoder processes the incoming traffic to reconstruct lost packets, recovering the original data. Each of the seven scenarios is repeated ten times for all the three traffic types (audio, video, and haptic), as the resulting 95% confidence were small.

### *F. Evaluation results*

In each test run, we record three KPMs for each type of traffic in a one-second interval. Consequently, we obtain multiple KPM results for each monitoring scenario. We start by calculating the mean of these results within each test run. The final mean value for each KPM is then obtained by averaging the results across ten independent test runs. These final mean values are presented in the plots.

*a) PDCP SDU Volume:* Fig. 3 presents the mean PDCP SDU volume results captured by *HawkVision* for the RLNC schemes (SW and SBC) in different types of traffic. In the baseline scenario, for all types of traffic, the mean PDCP SDU volume is low as expected, due to the absence of additional redundancy. Within SW and SBC, an increase in redundancy generally corresponds to a rise in the mean PDCP SDU volume. However, the extent of this increase depends on the traffic characteristics. In haptic traffic, we observe that a lower redundancy configuration of SBC, such as *SBC(8)*, produces a similar mean PDCP SDU volume to a high redundancy configuration of SW, such as *SW(8)*. This is due to haptic's small payload size (82 B), which amplifies the impact of SBC's per-packet overhead. The overhead from SBC and SW increases the packet size to 151 B and 109 B, respectively. Thus, despite the lower redundancy of *SBC(8)*, it contributes significantly to the mean PDCP SDU volume of haptic. In contrast, video's large payload size reduces the relative impact of RLNC's per-packet overhead, but redundancy from RLNC still adds significantly to the mean PDCP SDU volume.

*b) Total PRB Utilization:* Fig. 4 shows the mean PRB results collected by *HawkVision* for the different variants of SW and SBC in audio, haptic, and video. For all types of traffic, the lowest mean PRB is for the scenario without RLNC, as there is no redundancy. We observe that with the increasing level of redundancy in SW and SBC, the mean PRBs utilization increases for all traffic types. We note that PRB is the smallest unit of resources allocated (12 subcarriers $\times$ time slots) for data transmission. As expected, it can be underutilized for small packets like haptic with 82 B of payload, since the entire PRB is still allocated even if only a portion is used. It is also worth noting that the large payload size and the low inter-packet delay of video traffic contributed to significantly higher resource block usage, with a peak of 7863 PRB obtained for *SBC(32)*.

*c) UE Throughput:* Fig. 5 shows the variations in the mean UE throughput recorded for audio, haptic, and video traffic under different redundancy levels for both SW and SBC. As anticipated, the baseline scenario, which does not

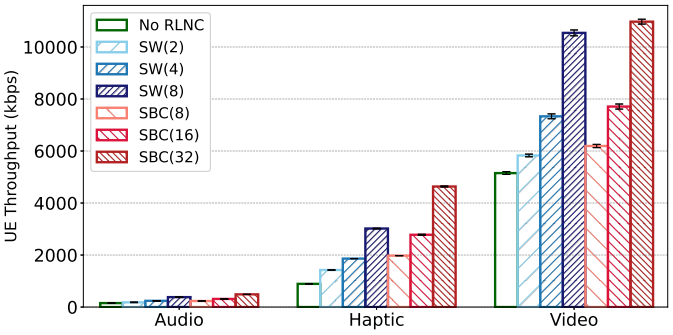


Fig. 5. Mean UE throughput monitored for various traffic

involve redundancy, produces the lowest mean UE throughput for all traffic types. *HawkVision* shows that the RLNC schemes have a greater impact on the mean UE throughput than on the mean PRB shown in Fig. 4, particularly for the case of small payload and high-frequency traffic such as haptic. As mentioned previously, haptic traffic can underutilise PRBs due its small size. In contrast, UE throughput captures the total number of bits transmitted, including redundancy and RLNC encoding overhead over a period of time.

In summary, integration of RLNC schemes such as SBC and SW improves reliability and introduces low latency [14], but also increases the use of radio resources, reflected in higher PDCP SDU volume, PRB utilization, and UE throughput, as observed through *HawkVision*. This overhead is expected to grow with higher settings, such as increased redundancy. We also note that the RLNC schemes with high redundancy settings, such as SBC(32), are only used in rare circumstances with extremely high packet loss.

## VI. Conclusion

We proposed *HawkVision*, a monitoring solution for RLNC operations in mobile networks, which leverages the flexible architecture of O-RAN. *HawkVision* consists of xApps running within the Near-RT RIC and RLNC components integrated in the user plane of the mobile network. Using *HawkVision*, we evaluated Systematic Block Code and Sliding-Window schemes on a practical testbed under various settings. We demonstrated that *HawkVision* can provide deep insights into radio resource costs related to RLNC by near-real-time monitoring of KPMs, such as PDCP SDU volume, PRB usage and UE throughput. These findings lay the groundwork for an adaptable, RLNC-based mobile system. The results provide insight into resource demands under varying conditions, enabling dynamic adjustments. The model is built on a Docker-based framework for easy deployment and integrates a bare-metal gNB and UE, combining software flexibility with real hardware interaction. While the current setup uses a single UE to evaluate encoder performance, the architecture is extensible and can be scaled to support multiple UEs for broader testing. Future use cases, such as autonomous aerial vehicles and remote surgical systems, can benefit from *HawkVision* by applying different RLNC schemes and simultaneously controlling their operations.

## Acknowledgment

This work was funded by the Federal Ministry of Research, Technology and Space of Germany (BMFTR) in the programme of ”Souverän. Digital. Vernetzt.” Joint project 6G-life, project identification number: 16KISK001. We also gratefully acknowledge funding from BMFTR under the grant 01|S23070 (for the Software Campus project NC5G). Additionally, this work was supported by the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany’s Excellence Strategy – EXC 2050/1 – Project ID 390696704 – Cluster of Excellence “Centre for Tactile Internet with Human-in-the-Loop” (CeTI) of Technische Universität Dresden.

## References

[1] A. Caruso *et al.*, “Adaptive 360° video streaming over a federated 6g network: Experimenting in-network computing for enhanced user experience,” in *IEEE IWNC*, 2024.
[2] O. Lhamo *et al.*, “Flexnc + recnet: Flexible network (re)coding in cloud-native 5g: Design and testbed measurements,” *IEEE TNSM*, 2025.
[3] Shahzad *et al.*, “RS-RLNC: A reinforcement learning-based selective random linear network coding framework for tactile internet,” *IEEE Access*, 2023.
[4] E. Tasdemir, “Computational complexity and delay reduction for rlnc single and multi-hop communications,” 2023.
[5] E. Tasdemir *et al.*, “Sparec: Sparse systematic rlnc recoding in multi-hop networks,” *IEEE Access*, 2021.
[6] S. Pandi *et al.*, “PACE: Redundancy engineering in RLNC for low-latency communication,” *IEEE Access*, 2017.
[7] M. A. Kamal *et al.*, “Resource allocation schemes for 5G network: A systematic review,” *Sensors*, 2021.
[8] B. Lai *et al.*, “Resource-efficient generative mobile edge networks in 6G era: Fundamentals, framework and case study,” *IEEE Wireless Communications*, 2024.
[9] S. Marinova *et al.*, “Intelligent O-RAN beyond 5G: Architecture, use cases, challenges, and opportunities,” *IEEE Access*, 2024.
[10] R. Schmidt *et al.*, “Flexric: An sdk for next-generation sd-rans,” in *ACM CoNEXT*, 2021.
[11] W. Mao *et al.*, “Network Coding for Integrated Access and Backhaul Wireless Networks,” in *WOCC*, 2020.
[12] T. Peng *et al.*, “Physical layer network coding in network mimo: A new design for 5g and beyond,” *IEEE TCOM*, 2019.
[13] D. Vukobratovic *et al.*, “Random Linear Network Coding for 5G Mobile Video Delivery,” *Information*, 2018.
[14] O. Lhamo *et al.*, “Improving reliability for cloud-native 5G and beyond using network coding,” in *IEEE NFV-SDN*, 2023.
[15] M. Polese *et al.*, “Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges,” *IEEE Communications Surveys & Tutorials*, 2023.
[16] O. T. Başaran *et al.*, “Deep autoencoder design for rf anomaly detection in 5G O-RAN near-RT RIC via xApps,” in *IEEE ICC Workshops*, 2023.
[17] R. Ferreira *et al.*, “Demo: Enhancing network performance based on 5g network function and slice load analysis,” in *IEEE WoWMoM*, 2023.
[18] “OAI full stack 5G-NR RF simulation with containers,” https://gitlab.eurecom.fr/oai/openairinterface5g/-/tree/develop/ci-scripts/yaml_files/5g_rfsimulator, 2024, accessed: 2024-11-27.
[19] O-RAN Alliance, “O-RAN-WG3.TS.E2SM-KPM-R004-v06.00: E2 Service Model (E2SM) KPM,” O-RAN Alliance, 2025.
[20] S. Wunderlich *et al.*, “Caterpillar RLNC (CRLNC): A practical finite sliding window RLNC approach,” *IEEE Access*, 2017.
[21] S. S. Nakkina *et al.*, “Performance benchmarking of the 5G NR PHY on the OAI codebase and USRP hardware,” in *IEEE WSA*, 2021.
[22] E. Tasdemir *et al.*, “FSW: Fulcrum sliding window coding for low-latency communication,” *IEEE Access*, 2022.
[23] S. Yi *et al.*, *Radio Protocols for LTE and LTE-Advanced*. Hoboken, NJ, USA: John Wiley & Sons, 2012.
[24] 3GPP, “TS 28.552 v18.8.0 - 5G; management and orchestration; 5G performance measurements,” 3GPP, 2024, release 18.
[25] “O-RAN KPM in OAI,” https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/openair2/E2AP/RAN_FUNCTION/O-RAN/ran_func_kpm_subs.c, accessed: Mar. 21, 2025.
[26] D. H. Morais, “5G NR overview,” in *5G NR, Wi-Fi 6, and Bluetooth LE 5*. Springer, Cham, 2023.