Suez Canal University
Faculty of Engineering
Department of Electrical Engineering
Electronics and communications program

Graduation Project
ENERGY SAVING USE CASE IN OPEN RAN

Prepared by

Omar tarek
Ziad Hossam
Tasneem amin
Eman tarek
Demond Wael
Yousef Ashraf
Mohamed Hesham
Fouad Yasser
Mohaned Hesham
Amr Fawzy
Amar Elsayed
Ziad Abdullah
Mina Akram

Supervised by:

Dr. Doaa Gamal **Dr. Mohamed Yousef**

Sponsored and mentored by

Eng: Aya Kamal **Eng: Mina Yonan** **Eng: Abdelrahman Khaled**

ACKNOWLEDGEMENTS

"الْحَمْدُ لِلَّهِ الَّذِي هَدَانَا لِهَذَا وَمَا كُنَّا لِنَهْتَدِيَ لَوْلا أَنْ هَدَانَا اللهُ"

In the Name of Allah, the Most Compassionate, the Most Merciful, we begin by expressing our deepest gratitude to Allah for His blessings, guidance, and mercy throughout our academic journey. Without His help, we could not have achieved this milestone in our lives.

We would like to extend our thanks to our two supervisors **Dr. Doaa Gamal** and **Dr. Mohamed Yousef** for continuous support throughout the project and the Guidance through all the tasks and telling us how to act properly and professionally Their encouragement and experience and expertise throughout the project has helped us Make it through and it was a big help to us in this graduation project

We also want to thank **Eng. Mina Yonan** for giving us the opportunity to be part of the mentorship from Orange this experience has greatly enhanced our knowledge about the telecommunications market and the Open Ran market and giving us the invaluable insights

Into the industry

A special thanks to Eng. Aya Kamal and Eng. Abdelrahman Khaled for the efforts and all the technical details they provide us with and the tons of questions that they handled with us the and the patience all this a very special thanks to them their guidance was a key stone in our project as they were responsible for the guidance of the part 1 and part 2 team

Table of Contents

1. List of Figures

2. List of Abbreviation

Abbreviation	Full Term
5G	Fifth Generation
AI	Artificial Intelligence
API	Application Programming Interface
BB	Baseband
BF	Beamforming
BS	Base Station
COOS	Cell On/Off Switching
CPU	Central Processing Unit
DL	Downlink
DU	Distributed Unit
KPI	Key Performance Indicator
LTE	Long-Term Evolution
MAC	Medium Access Control
MIMO	Multiple Input Multiple Output
ML	Machine Learning
NF	Network Function
NR	New Radio
ns-3	Network Simulator 3
O-CU	O-RAN Central Unit
O-DU	O-RAN Distributed Unit
O-RAN	Open Radio Access Network
O-RU	O-RAN Radio Unit
PHY	Physical Layer
PM	Precoding Matrix
QoS	Quality of Service
RAN	Radio Access Network
RCR	RF Channel Reconfiguration
RF	Radio Frequency
RIC	RAN Intelligent Controller
RL	Reinforcement Learning
RT	Real-Time
RU	Radio Unit
SDN	Software Defined Networking
SRS	Sounding Reference Signal
TCO	Total Cost of Ownership
UE	User Equipment
UL	Uplink
VM	Virtual Machine
VRAN	Virtualized Radio Access Network
NR	**New radio**

NS3	Network simulator 3
FDD	Frequency division duplexing
TDD	Time division duplexing
PMI	Precoding matrix indicator
RI	Rank indicator
CQI	Channel quality indicator
UE	User equipment
Gnb	Gnode b (base station in 5G)
BW	Bandwidth
RB	Resource block
TX	Transmit
RX	Receive
ES	Energy saving

Abstract

The evolution of mobile networks from 2G through 5G and toward 6G has significantly increased data demand, infrastructure complexity, and energy consumption. As networks become denser and more sophisticated, the need for sustainable and energy-efficient solutions has become critical. This project addresses that challenge by exploring a dynamic, AI-driven energy-saving use case within the Open Radio Access Network (O-RAN) framework. It focuses on using agentic artificial intelligence to intelligently manage and reduce power consumption in next-generation mobile networks without compromising network performance or user experience.

By leveraging key technologies such as Massive MIMO, beamforming, and RF channel reconfiguration, the project aims to reduce unnecessary energy use while maintaining Quality of Service (QoS). A simulation environment was developed using the ns-3 network simulator and the 5G-LENA module, enabling detailed experimentation with PHY-layer antenna port power control. The results demonstrated up to 17% power savings during low-traffic periods, showcasing the potential of intelligent energy adaptation strategies in real-world scenarios.

Additionally, the project incorporates cloud-native and virtualized principles inherent in the O-RAN architecture. These principles allow for scalable, flexible deployment of energy-saving algorithms and seamless integration with AI-powered control loops. The agentic AI component enables real-time, autonomous decision-making, dynamically adjusting power based on traffic demand and environmental conditions.

This research contributes to the broader vision of sustainable network design, aligning with global efforts to reduce carbon footprints while supporting the demands of modern communication systems. It serves as a foundation for developing more intelligent, eco-friendly 5G and future 6G networks.

Introduction

- Any mobile communication network consists of three primary components: the Mobile Station, the Radio Access Network (RAN), and the Core Network. Among these, the RAN plays a critical role in providing wireless connectivity, managing resources, and ensuring the delivery of services with high quality and low latency.
- Over the generations, RAN architecture evolved from fully integrated and vendor-locked designs in 2G/3G, to partially centralized structures in 4G, and now towards open, disaggregated architectures in 5G and beyond with O-RAN. Traditional RANs suffered from limited flexibility and high operational costs, while O-RAN introduces a modular, open-interface structure, decoupling hardware from software and enabling multi-vendor interoperability, scalability, and cost efficiency.
- The flexibility of O-RAN, combined with its support for virtualization and cloud-native deployment, enables dynamic resource allocation and management, making it ideal for implementing energy-saving strategies. This project leverages these advantages to explore energy-efficient use cases within O-RAN using techniques such as Massive MIMO, beamforming, and RF Channel Reconfiguration (RCR), integrated with agentic AI to dynamically control antenna port power consumption.
- Using the ns-3 simulator and the 5G Lena module, the project implements and tests dynamic antenna port power control under various simulated traffic conditions, validating the feasibility of achieving practical energy savings while maintaining network performance. By integrating AI-driven automation with the programmable nature of O-RAN, this project contributes to advancing the sustainability and efficiency of next-generation mobile networks

Brief History

- The evolution of mobile networks from 2G to 5G and beyond has created an urgent need for sustainable, energy-efficient solutions while maintaining high network performance and cost-effectiveness. This project investigates an energy-saving use case within the Open Radio Access Network (O-RAN) framework using agentic AI for dynamic power management in next-generation mobile networks. By leveraging advanced techniques such as Massive MIMO, beamforming, and RF Channel Reconfiguration, the project targets the reduction of power consumption while ensuring Quality of Service (QoS).
- A simulation environment was developed using the ns-3 simulator and the 5G Lena module to implement and test dynamic antenna port power control within the PHY layer, demonstrating up to 17% energy savings during low-traffic periods. The

Open Ran Market

the global open Ran market size for estimated to be about 4.5B dollars in 2024 and is projected to reach about 20B dollars by 2030, growing at a rate of 25.6 %

An open radio network or O-RAN is a nonproprietary version of the radio access network RAN that allows the cooperation between the cellular network equipment between the different vendors out there

Open Ran enables operators to choose the hardware and the software solutions from various vendors which can lead to reduced costs and more flexibility in the operational costs and the equipment costs

In February 2018 China , AT&T Inc., Deutsche Telekom, Orange, and NTT DOCOMO, founded the O-RAN ALLIANCE.

The alliance aims to reshape the Ran industry toward more open and intelligent ways to

Make mobile networks the forum set O-RAN specifications that enable a more vibrant and competitive RAN supplier ecosystem with quicker innovations to improve user experience. The O-RAN ALLIANCE plays a central role in developing the Open Radio Access Network (O-RAN) market. Open RAN enables operators to choose different hardware and software components from various vendors per their budgetary

constraints. This freedom of choice regarding components and vendors translates into a cost-effective and flexible ecosystem compared to the traditional RAN architecture.

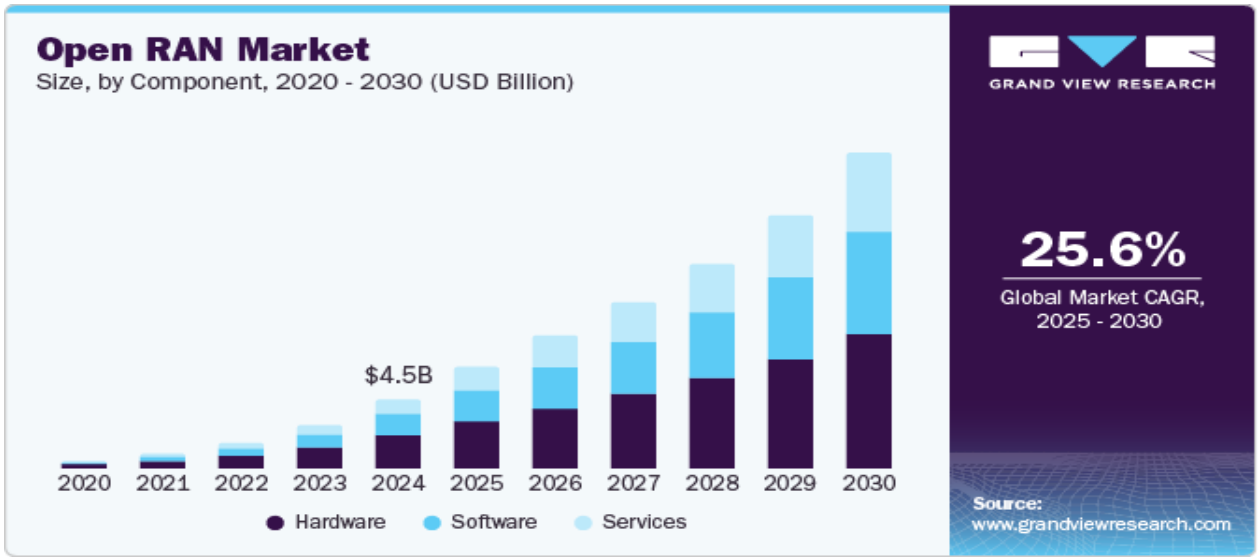


fig. 1-1. Open Ran market

Regulatory support and geopolitical considerations are also driving the Open RAN market forward. Governments, particularly in the U.S. and Europe, are increasingly viewing Open RAN as a strategic approach to enhancing network security and reducing dependency on specific vendors, especially considering concerns about potential security risks associated with certain foreign suppliers. This endorsement from regulators not only promotes competition but also encourages local and smaller technology vendors to enter the telecom space, fostering innovation and strengthening regional technology ecosystems. As a result,

Open RAN is emerging as a crucial component of next-generation telecommunications infrastructure, aligning with broader priorities for security, cost-effectiveness, and flexibility in global markets. Furthermore, the collaboration between various equipment providers and network operators allowed by O-RAN makes it easier for operators to deploy new features and services. The open standards allow for more flexibility and customization, thus fostering innovation in the market. The open RAN is still in its nascent stages of development and still requires some standardization efforts. Network standardization is important to ensure that different components from different vendors can work together seamlessly. Security concerns arise in the market as new technology introduces new security risks. Open RAN networks need to be more secure to protect customer data. However, as the market matures over time, these hurdles will be overcome with the evolving technology and ecosystem of the O-RAN.

North America open RAN market dominated the industry and accounted for the largest share of 41.2% in 2024. The regional growth can be attributed to several factors, including the presence of prominent key players, such as AT&T, Inc., and the region's technological inclination toward early adoption of advanced solutions. Moreover, the region is witnessing significant investment in telecommunication infrastructure, creating significant growth opportunities for adopting O-RAN. The convergence of these factors has propelled North America to the forefront of Open RAN expansion, fostering the regional market's growth and development in the market.

fig. 1-2. open Ran market share

Europe Open Market Trends

The Europe Open RAN market was identified as a lucrative region in 2024. In Europe, the Open RAN market is experiencing rapid expansion fueled by the continent's push towards adoption of 5G and the desire for more flexible network architectures. Governments across Europe are investing in improving their networking infrastructure. For instance, the European Commission committed public funding of more than EUR 700 million through the Horizon 2020 program to support 5G opportunities in the region. The investments are necessary to support the 5G traffic volume expected by 2025. Such initiatives are expected to drive the market's growth from 2024 to 2030.

Many countries and companies race to deploy and introduce the open ran standard as it will decrease the costs of the operational and the equipment costs based on the collaboration between the companies . There are some companies that already begin trails and deployments

All over the world as shown in this graph

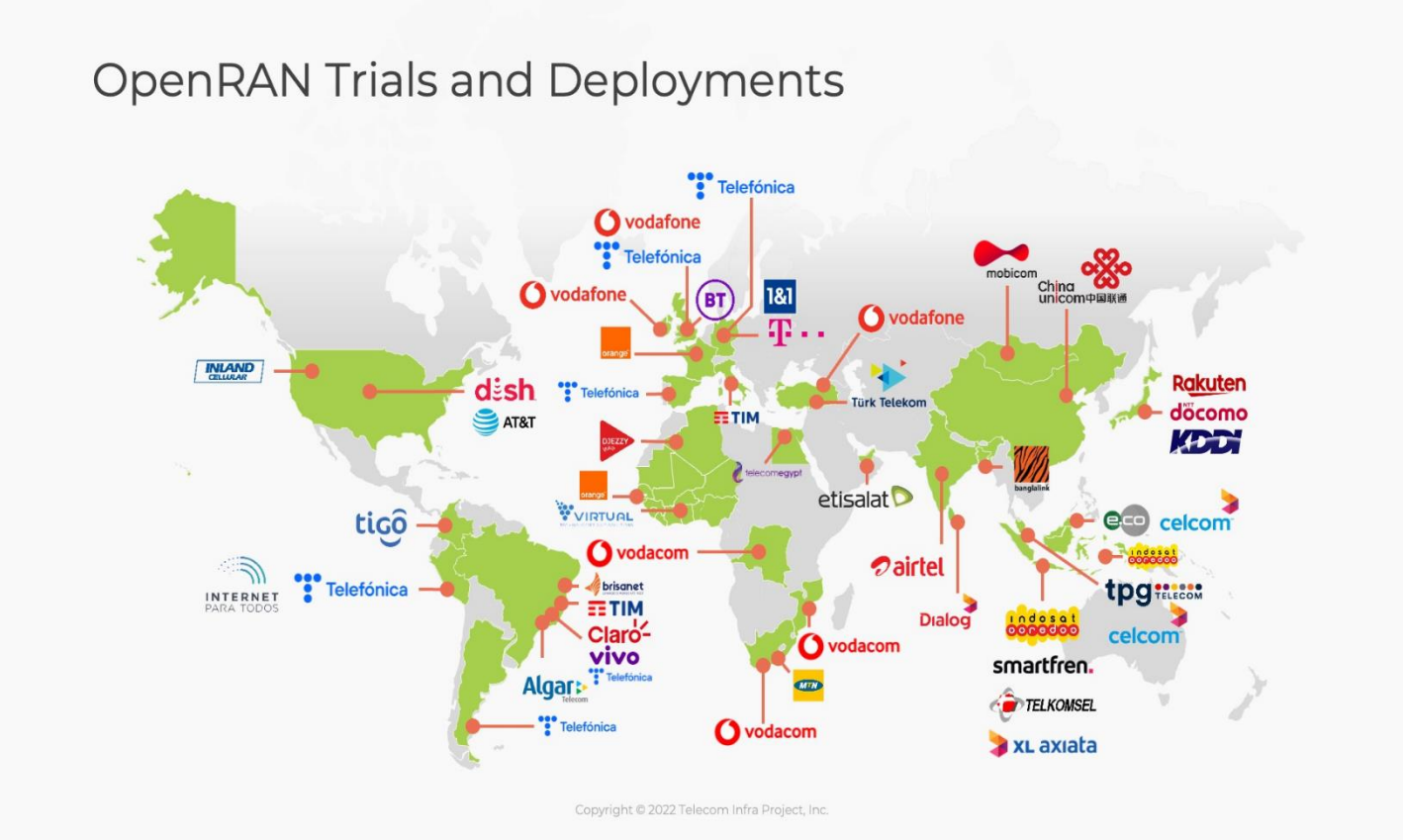


fig. 1-3.Open Ran deployment Map

Chapter 2 project overview

2.1 RF Channel Reconfiguration use case and energy saving

Energy Efficiency (EE) has become a major focus in modern Radio Access Networks (RAN), especially with the high throughput demands of 5G and beyond. To reduce energy use while maintaining Quality of Service (QoS), dynamic control of hardware and resources based on traffic demand is essential. Energy Saving (ES) mechanisms now span all parts of the RAN—from hardware to software and orchestration—and have been addressed in 5G standards starting with 3GPP Release 18.

Traditional RAN systems have limited energy efficiency due to fixed hardware use, while Open RAN offers better flexibility with virtualized components and open interfaces. The O-RAN architecture introduces intelligent control through RAN Intelligent Controllers (RICs): the Non-Real Time RIC for long-term control and the Near-Real Time RIC for quicker adjustments. These RICs use AI-powered applications (Rapps and xApps) to implement energy-saving strategies such as micro sleeps and hardware reconfiguration.

These use cases are key components in making 5G and beyond networks more **energy efficient** while maintaining **high Quality of Service (QoS)**

1- Cell On/Off Switching (COOS)

2- RF Channel Reconfiguration (RCR)

2.1.1Cell On/Off Switching (COOS):

- This technique **turns off underutilized cells** (or base stations) during low traffic periods (e.g., nighttime) to save energy.
- It helps **reduce power consumption** without affecting network performance, by dynamically adapting to traffic demand.

2.1.2RF Channel Reconfiguration (RCR):

- Also known as **antenna selection** in Massive MIMO systems.
- It involves **activating only a subset of antennas** based on current traffic or user distribution.

- This saves energy while maintaining coverage and capacity, especially in **dense antenna deployments**.

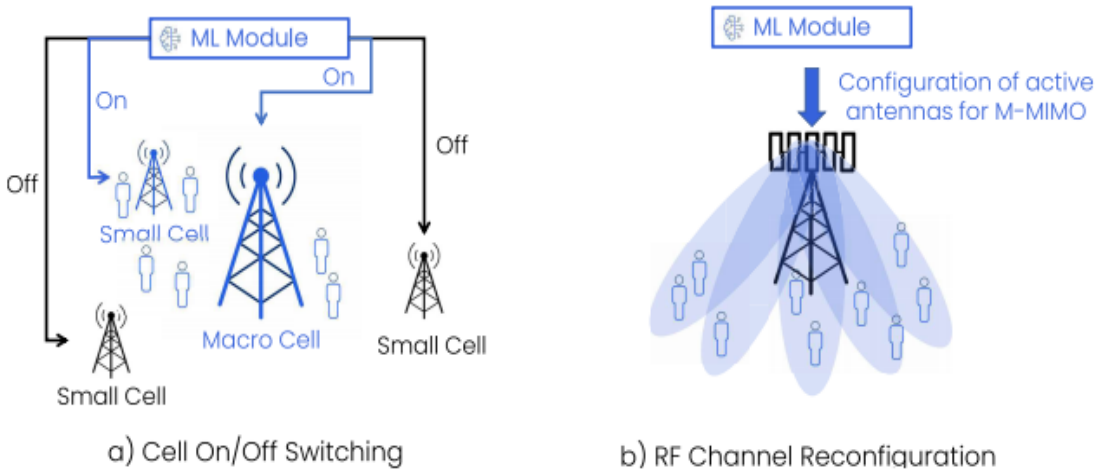


fig. 2-1. Ml assisted cell on and off

2.2 MIMO And SU-MIMO

MIMO is a family of technologies that uses multiple antennas at both transmitter and receiver sides to improve communication performance. A key method is Spatial Multiplexing (SM), where multiple data streams (called layers in 3GPP) are transmitted either to a single user (SU-MIMO) or distributed across multiple users (MU-MIMO), increasing spectral efficiency.

Massive MIMO is a promising technology for the next generation of wireless communication systems. It has the potential to improve the performance of these systems in several ways and is currently being studied by a number of research groups around the world.

At millimeter wave (mm Wave) frequencies, beamforming is crucial due to high pathloss and signal blockage. It helps direct energy to the user, improving performance and reducing interference. While spatial multiplexing can further increase data rates, mm Wave systems are limited by the high cost and power usage of RF chains, which restricts the number of simultaneous data streams.

Massive MIMO is a promising technology for the next generation of wireless communication systems. It has the potential to improve the performance of these systems in several ways and is currently being studied by a number of research groups around the world.

Massive MIMO, with its large antenna arrays, offers significant benefits for wireless communication, primarily by increasing network capacity, improving spectral efficiency, and enhancing user experience. It allows for higher data rates, better coverage, and more reliable connections, especially in densely populated areas

1. Increased Network Capacity:

- Massive MIMO can serve a much larger number of users simultaneously compared to traditional MIMO systems.
- This is achieved by using spatial multiplexing, where multiple data streams are transmitted and received using different antennas, essentially creating parallel channels.
- In densely populated areas, where many users are competing for network resources, massive MIMO can significantly alleviate congestion and improve overall capacity.

2. Enhanced Spectral Efficiency:

- Massive MIMO improves spectral efficiency, which is the amount of data that can be transmitted over a given frequency bandwidth.
- By using beamforming and spatial multiplexing, it can transmit more data within the same bandwidth, making more efficient use of the available spectrum.
- This leads to higher data rates and improved performance for users.

3. Improved User Experience:

- Massive MIMO provides better signal strength and coverage, especially at the cell edges where signals tend to be weaker.
- Beamforming technology, a key component of massive MIMO, focuses the radio signal directly towards the user, minimizing interference and maximizing signal strength.

- This results in a more consistent and reliable connection for users, even in challenging environments.

4. Energy Efficiency:

- While massive MIMO uses more antennas, it can also lead to improved energy efficiency.
- By focusing the signal on the user and reducing interference, massive MIMO can achieve higher data rates with the same or even lower power consumption.
- This is particularly important for mobile devices, where battery life is a key consideration.

5. Reduced Interference:

- Beamforming and spatial multiplexing in massive MIMO help to reduce interference between users.
- By directing signals specifically to individual users, it minimizes the impact of signals intended for other users on a user's connection.

Fig.2 Without MIMO

fig.3 With MIMO

fig. 2-2.How 4x4 MIMO Improve capacity and coverage

SU-MIMO vs. MU-MIMO

Spatial multiplexing refers to the technique of multiplexing several data streams, or layers, on the same time-frequency symbol. The multiplexed data streams can all go to the same device or to different devices. Cases in which all the layers belong to the same device are referred to as single-user MIMO (SU-MIMO), while cases that involve spatial multiplexing of multiple devices are called multi-user MIMO (MU-MIMO). Spatial multiplexing can increase spectral efficiency, which translates into increased user throughput and network capacity .

Beamforming and SU-MIMO are central to Massive MIMO. The ability of beamforming to increase the received signal level while not increasing the average interference level is key to obtaining high performance. Substantial

Key Parameter	Single-user **MIMO**	Multi-user **MIMO**
Communication Type	Point to point	Point to multipoint
Coverage	Less	Higher
Average cell throughput	Less	Higher
Average cell edge throughput	Higher	Less
Propagation channel	Less immunes against the ill-behavior of propagation channel	More immunes against the ill-behavior of propagating channel
Antenna correlation (Small spacing-ULA)	Work better in low Correlation	Work better in high Correlation
Mobility	Preferable	Not preferable
Channel quality	Less sensitive	More sensitive
Heavily loaded Cell	Not preferable	Preferable
Complexity	Less	Higher
Bandwidth	Used more BW	Better usage of the BW available

fig. 2-3.Comparison of SU-MIMO and MU-MIMO System

beamforming gains can be achieved in a wide range of situations, regardless of DL/UL, traffic load, or if the user is in a good or bad spot. Coverage, capacity, and user throughput are generally improved

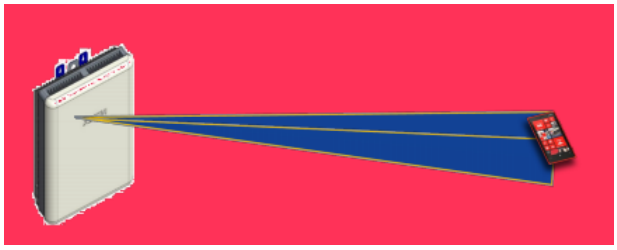


fig. 2-4. single user mimo

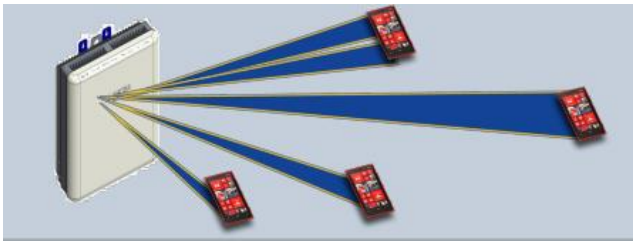


fig. 2-5. multiple user MIMO

2.2.1 MIMO fundamentals

. A-Spatial multiplexing principle

• MIMO = Multiple Input Multiple Output

• Different symbols are sent by transmit antennas in the same time and frequency

• As symbols propagate over the wireless channel, their phase and amplitude changes according to the channel coefficient (complex value)

• Channel between each pair of the TX and RX antennas

is different due to different propagation conditions

• Receiver needs only to solve a set of equations. Channel coefficients are known from reference signals, so the only unknown is the transmitted symbols

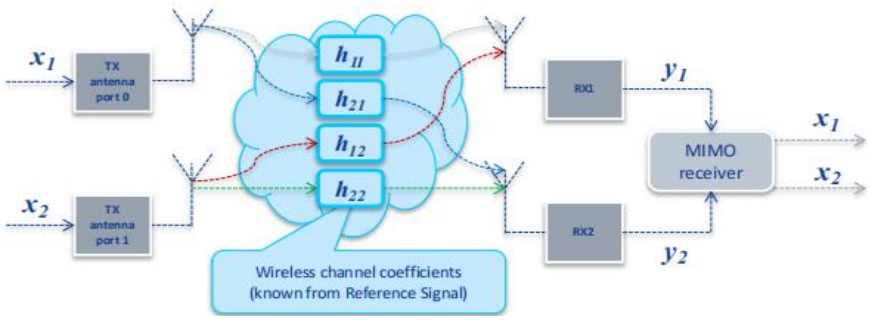


fig. 2-6. How to find channel coefficients

B-Precoding

Precoding is a technique used in MIMO systems to adjust the transmitted signal before it is sent over the wireless channel. It involves applying a matrix, known as the precoding matrix, to the data stream before it is transmitted. This matrix essentially shapes the transmitted signal to compensate for channel impairments and improve reception at the receiver.

Why use Precoding?

- Enhanced Signal Quality
- Increased Data Rates
- Improved Spectral Efficiency
- Reduced Interference

C-Feedback

A feedback matrix refers to the information about the channel that is fed back from the receiver to the transmitter to enable efficient communication. This feedback is crucial for techniques like beamforming and precoding, allowing the transmitter to adapt its signals to the channel conditions. The feedback matrix can represent various aspects of the channel, including channel state information (CSI), covariance matrices, or even precoding matrices themselves.

Purpose of Feedback

- Channel Adaptation
- Precoding
- Beamforming

Types of Feedback Matrices

- Channel State Information (CSI)
- Covariance Matrices
- Precoding Matrices

D- Codebook

it would many different things in different situation, but the meaning of Codebook under the context of CSI-RS is a set of Precoders (a set of Precoding Matrix).

Putting it another way, Codebook is a kind of matrix (a matrix having complex value elements) that transform the data bit (PDSCH) to another set of data that maps to each antenna port.

Types of Codebooks

- Type I
- Type II

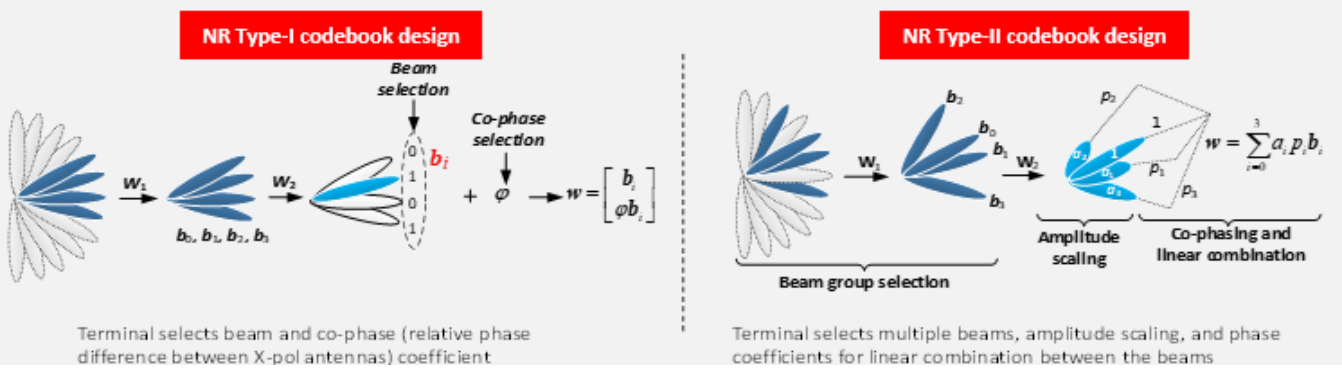


fig. 2-7.codebook

DL Closed Loop Transmission

- NB transmits CSI-RS and UE reports CSI-RS feedback (CRI-RI-PMI-CQI)
- NB selects the rank and the precoding matrix according to the UE's CSI feedback

2.3. Massive MIMO Features and Beamforming

Evolution from MIMO to Massive MIMO

The evolution from MIMO to Massive MIMO represents a significant leap in wireless technology. Traditional MIMO provided a foundation for spatial multiplexing and improved signal quality. Massive MIMO builds upon this foundation by dramatically increasing the number of antennas, enabling more sophisticated beamforming techniques and supporting a much larger number of users and devices.

Massive MIMO Features and Beamforming On 5G

5G New Radio (NR) has been designed to fully support Massive MIMO as an active technology from the start. The vastly increased coverage, capacity, and user throughput that Massive MIMO provides has quickly made it a natural and essential component of cellular network deployment

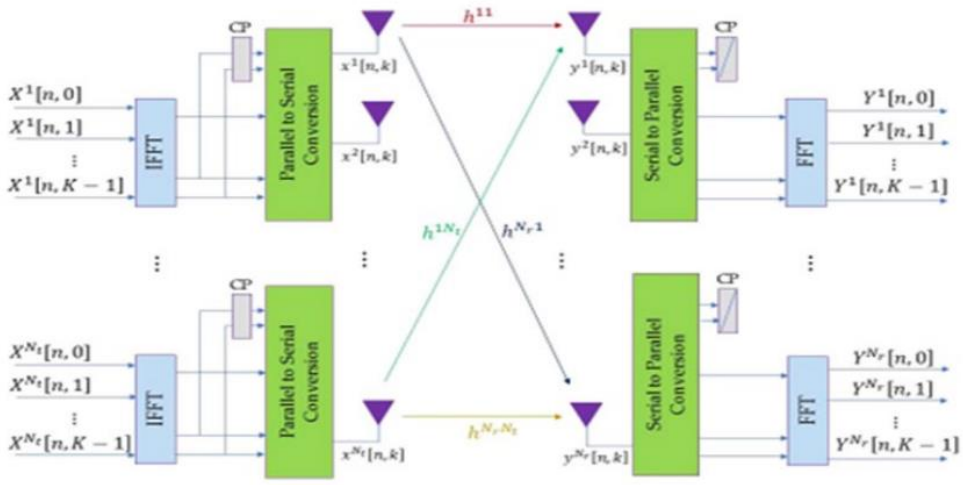


fig. 2-8.Massive MIMO

Massive MIMO (multiple-input, multiple-output) radios are the leading radio solution for new 5G deployments on mid-band TDD spectrum. The ability to use a larger number of radio chains – 16-64, for example, compared with the 2-8 typically used in conventional radio solutions – makes it possible for communication service providers (CSPs) to benefit significantly from multi-antenna techniques. Although Massive MIMO is still a relatively new technology – the term was first coined in academia about a decade ago with the first commercial solutions hitting the market about five years later – the technology has matured rapidly, enabling the creation of cost-efficient solutions that are both small and light in weight. Most of the Massive MIMO solutions on the market today have already gone through multiple hardware generations, often with optimized application-specific integrated circuits for beamforming and physical layer processing. They are available in a multitude of different models optimized for a wide variety of deployment scenarios [1,2,3].

Beamforming

Beamforming is a signal processing technique used to focus a signal in a specific direction, improving the signal strength and reducing interference from other directions. It works by combining signals from multiple antennas, carefully adjusting their phases and amplitudes to create a concentrated "beam" of signal.

Beamforming is a technique by which the mean beam of the antenna arrays can be steered in a specific direction with maximum directivity and power and almost no power in other directions.

Unless a physical object blocks the transmission, electromagnetic waves radiate in all directions from a single antenna. Multiple antennas placed near together broadcast the same signal at slightly different times in order to concentrate the signal and create a targeted electromagnetic energy beam. Interference caused by the overlapping waves will sometimes be beneficial (making the signal stronger).

Multiple antennas are used in beamforming to control the direction of a wavefront by controlling the magnitude and phase of individual antenna signals in the MIMO array.

The individual antennas in a MIMO array are spaced at least ½ wavelength apart. With Analog beamforming, the signal phases of individual antenna elements are adjusted in the RF domain; Analog phase shifters are used to steer the antenna array’s signal.

Beam steering

Beam steering is achieved by changing the phase of the input signal on all radiating elements. Phase shifting allows the signal to be targeted at a specific receiver. An antenna can employ radiating elements with a common frequency to steer a single beam in a specific direction. Different frequency beams can also be steered in different directions to serve different users. The direction a signal is sent in is calculated dynamically by the base station as the endpoint moves, effectively tracking the user. If a beam cannot track a user, the endpoint may switch to a different beam.

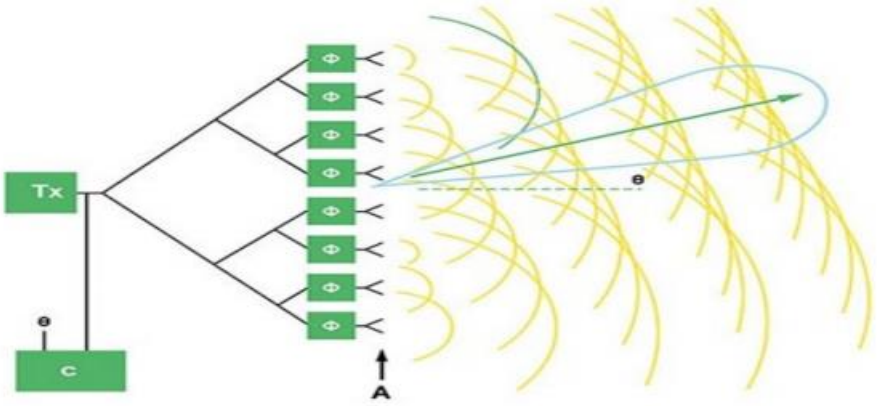


fig. 2-9.Beam_Steering

Benefits

- Increased Data Rates: By focusing on the signal, beamforming allows for faster data transmission.
- Increased Data Rates: By focusing on the signal, beamforming allows for faster data transmission.
- Increased Data Rates: By focusing on the signal, beamforming allows for faster data transmission.
- Increased Data Rates: By focusing on the signal, beamforming allows for faster data transmission.

Beamforming Types

Beamforming techniques can be grouped into three categories: **analog**, **digital** and **hybrid**.

Analog beamforming

Analog beamforming uses a single common RF source split among multiple antenna elements. The beam is controlled by adjusting analog phase shifters along the RF path. This is like the phased-array electrical down tilt discussed earlier.

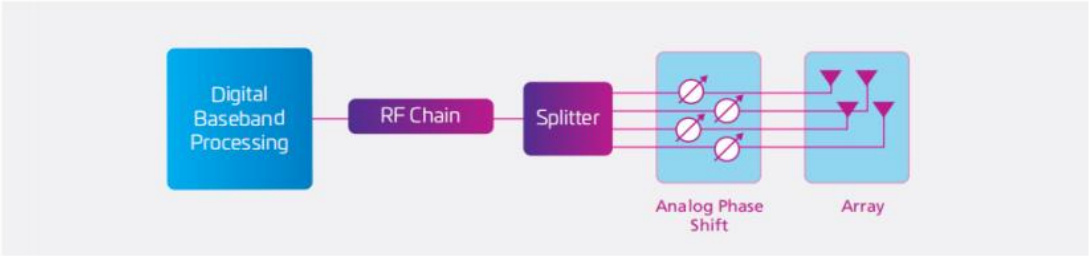


fig. 2-10.analog beamforming

Digital beamforming

In digital beamforming, each antenna has a dedicated RF signal and path. Phases and amplitudes are digitally controlled by baseband processing. Digital beamforming provides the best beam control; however, it suffers high power consumption and signaling overheads

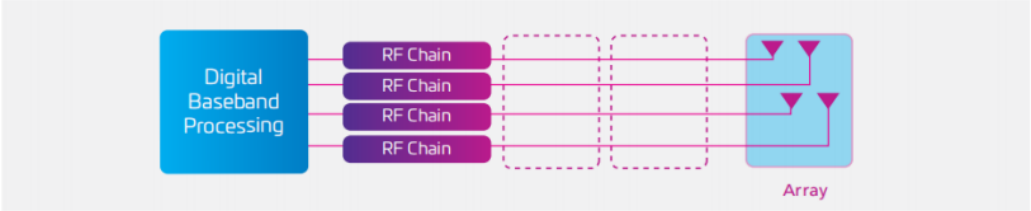


fig. 2-11.digital_beamforming

Hybrid beamforming

A compromise between the flexibility of digital beamforming and the lower cost and power consumption of analogue is provided by hybrid beamforming where analogue beamforming is carried out in the RF stage and digital beamforming is carried out in the baseband. The use of hybrid beamforming in large-scale mmWave antenna arrays is acknowledged as a cost-effective option, and several architectures are being researched for gNB (5G base station) implementations. These architectures can be roughly classified as completely connected—where each RF chain is connected to every antenna—and sub-

connected—where each RF chain is connected to a subset of antenna elements. Each architecture strives to provide performance that comes the closest to full digital beamforming while reducing the complexity of the hardware and signal processing.

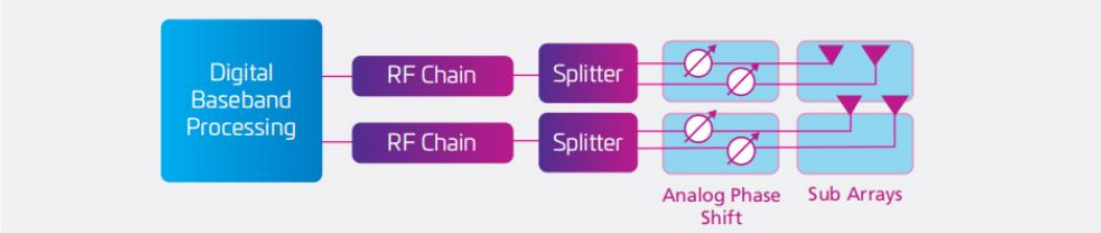


fig. 15. hybrid beamforming

2.4. PHYSICAL Layer of 5G-NR

5G NR PHY is designed to meet the main 5G use cases, namely eMBB, mMTC and URLLC. While it's an evolution of LTE PHY, there are many aspects that are unique to 5G NR PHY.

PHY layer sits at the bottom of the 5G NR protocol stack, interfacing to MAC sublayer higher up via transport channels. It provides its services to MAC and is configured by RRC. PHY supports downlink (Gnb-to-UE), uplink (UE-to-Gnb) and side link (UE-to-UE) communications.

Some of the main features include a wide spectrum from sub-GHz bands to mm Wave bands, an OFDM-based air interface, scalable numerology, deployments from indoor picocells to outdoor macrocells, FDD and TDD support, flexible and self-contained slot structure, modulation up to 256QAM, polar and LDPC

codes, Hybrid-ARQ (HARQ), bandwidth parts, CORESETs, beamforming, and massive MIMO.

breakdown of its key functions

- Error Detection & Correction: Acts as a vigilant inspector, identifying and correcting errors that might occur during transmission using powerful Forward Error Correction (FEC) encoding/decoding. Think of it like adding backup copies and checksums to your data for error-free delivery.
- Rate Matching & Channel Mapping: Adjusts the data rate to precisely fit the capacity of available physical channels and maps it efficiently, like a skilled architect optimizing space on the highway.
- Modulation & Demodulation: Transforms digital data into radio signals for transmission and back again upon reception, acting as the translator between the digital and physical worlds.
- Frequency & Time Synchronization: Ensures all devices are in perfect sync, like an orchestra conductor coordinating musicians, minimizing interference and maximizing efficiency
- MIMO Antenna Processing: Utilizes multiple antennas to create stronger, more reliable signals, comparable to having multiple lanes on the highway to increase traffic flow
- Transmit Diversity: Sends the same data through multiple antennas, like building redundant routes, to ensure it reaches its destination even if one path encounters obstacles.
- Beamforming: Focuses radio signals on specific users, akin to directing headlights for clear communication and reduced interference.
- Radio Characteristics Measurements: Continuously monitors channel conditions (signal strength, interference) and reports them to higher layers, like sending real-time traffic updates to optimize data flow

2.4.2 Media Access Control (MAC) layer

The 5G Media Access Control (MAC) layer, residing in Layer 2, plays a pivotal role in managing radio resources and ensuring efficient communication. Here are some key pointers to understand its functions:

- Beam Management: Directs radio signals towards your device for improved performance.
- Padding: Adjusts data size to match transmission requirements, further optimizing resource utilization.
- Mapping and Multiplexing: Translates different data types and combines them for efficient transmission.
- Scheduling: Determines which devices transmit when, optimizing resource allocation.
- HARQ Error Correction: Ensures reliable data delivery by retransmitting lost packets.
- Priority Handling: Prioritizes critical data (e.g., voice calls) for a seamless experience.
- Transport Format Selection: Chooses the most efficient way to pack data based on channel conditions.

Chapter 3 RF Reconfiguration

3.1 Brief history

First thing as the part 1 team was the working on the use case of energy saving and how to implement it on the NS3 and the 5G Lena module

First of all, what is NS3 and what is it used for

NS3 for short is a network simulator which is open source used in the academic and the industry research it is used for the simulation of the internet systems and network technologies like Wi-Fi , LTE / 5G it is developed from scratch using C++ What is so important about ns3 in this project ns3 is the simulation environment that we talked about earlier that doesn't have any ability to simulate the use case of the energy saving so to be able to use it as our base we needed to add a module that will help us in this matter and that is the 5G Lena module

The 5G Lena Module often called Nr in ns-3 is an official plugin for simulating NR networks

Within the ns-3 environment it's the successor of LTE Lena module which we can use to simulate the real world 5G performance. The reason we chose it was the ability to support the MIMO architecture and the beamforming and with configurable antenna arrays and the support for the sub-6 and the mm Wave frequencies and the FDD and the TDD operation,

Also, MIMO support: early versions allowed 2 streams (dual-polarized), while later releases added full MIMO implementations, including PMI/RI/CQI feedback, MMSE-IRC receivers, sub-band CSI-RS, dynamic rank adaptation, and more, this was an ultimate reason for choosing the plugin for simulating the use case of course the problem also with this that the

5G Lena module didn’t support the energy saving use case we need to alter and change the source files of the 5G Lena module that was that real issue and the challenge behind this project with the lack of resources and the lack of people that talk about the use case the lack of resources was the ultimate drawback of this for a sneak peek for what we did

The idea was simple you have a number of antenna elements, each of them consume a certain amount of power. Your job is to create a function that closes the power of each and certain

Antenna elements to conserve power as best as you can

So simple, right?

Well, it wasn’t as simple as digging in the physical layer of the base station and MIMO architecture that wasn’t the way base station gives power to the elements there way a lot of steps that were done before giving power to a certain antenna element in the simulation

Environment

We will cover all the steps we have taken to be able to reduce the power of the antenna using the MIMO configuration and we will show how we used a different approach to limit the power of the antenna using something in the PHY layer called precoding matrix

The first step to do is the installation of the ns3 and the NR Lena module

Which can be found in this link: https://cttc-lena.gitlab.io/nr/html/

Make sure when using this tutorial, we should use the NR Version 3.3 and NS3 version 3.42 for the outmost compactly with the changes that we made, we also recommend the using of ubuntu 20.04 and install GCC C++ any version you like before hand

You will see this in the terminal as the machine makes the build of the Nr and the NS3

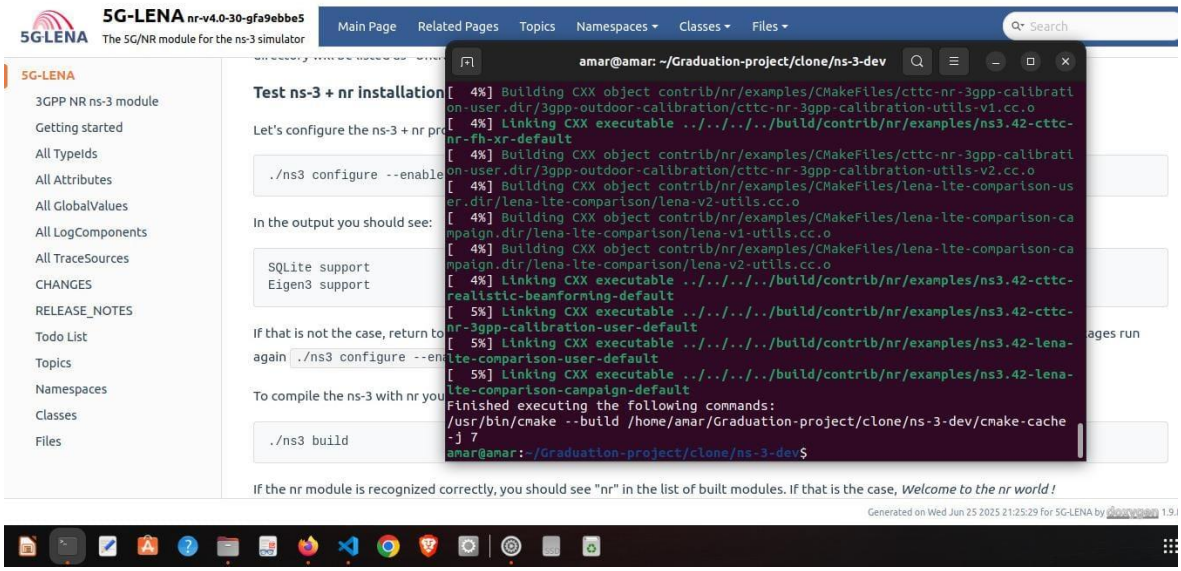
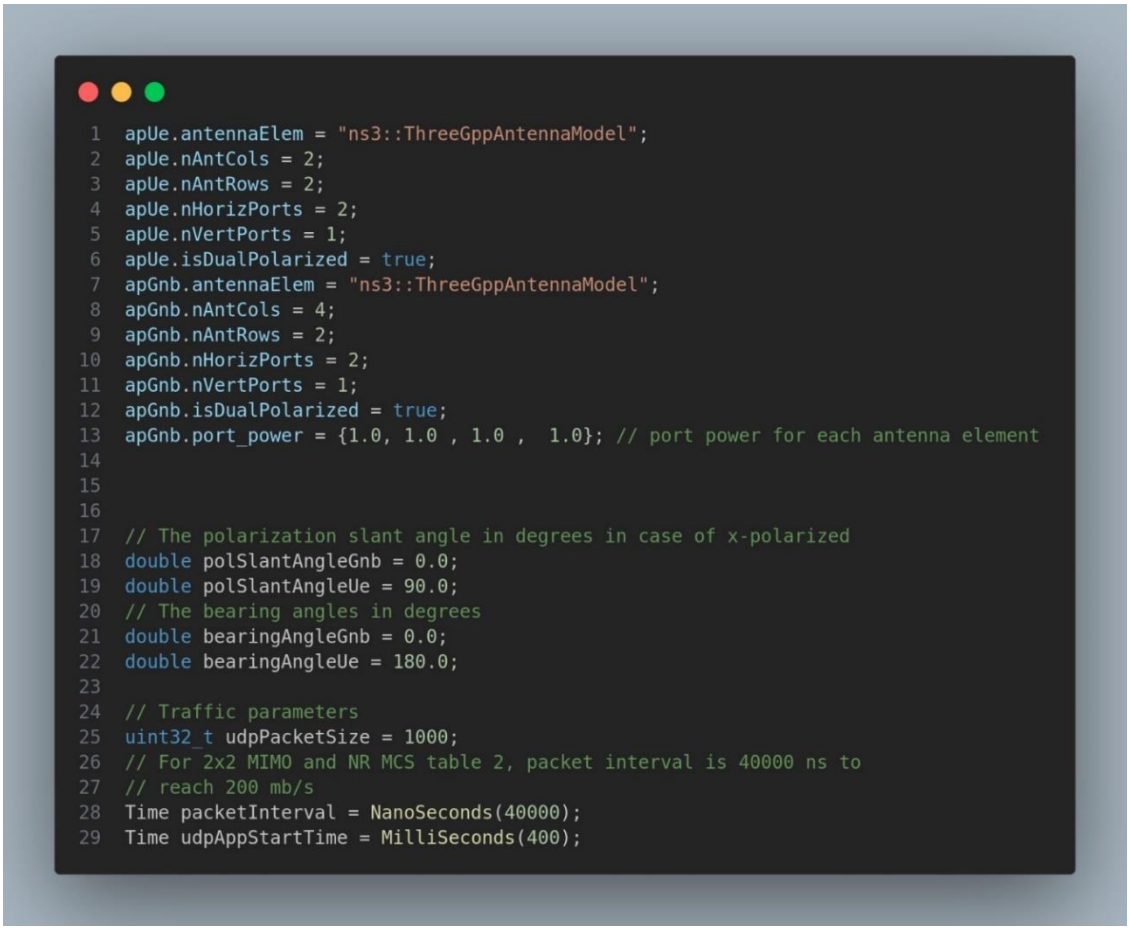


fig. 3-1. 5G Lena install process

After that we are ready to simulate some of the most popular scenarios in the Nr Module which is the cttc-nr-Mimo-demo
this scenario is the main scenario that we worked with and developing the source code and files for

That is the result of running the scenario in the terminal the scenario is just one Gnode B and one UE for short one base station and one user that the scenario is simulated with a chosen distance we can choose in the scenario



```
apUe.antennaElem = "ns3::ThreeGppAntennaModel";
apUe.nAntCols = 2;
apUe.nAntRows = 2;
apUe.nHorizPorts = 2;
apUe.nVertPorts = 1;
apUe.isDualPolarized = true;
apGnb.antennaElem = "ns3::ThreeGppAntennaModel";
apGnb.nAntCols = 4;
apGnb.nAntRows = 2;
apGnb.nHorizPorts = 2;
apGnb.nVertPorts = 1;
apGnb.isDualPolarized = true;
apGnb.port_power = {1.0, 1.0 , 1.0 ,  1.0}; // port power for each antenna element



// The polarization slant angle in degrees in case of x-polarized
double polSlantAngleGnb = 0.0;
double polSlantAngleUe = 90.0;
// The bearing angles in degrees
double bearingAngleGnb = 0.0;
double bearingAngleUe = 180.0;

// Traffic parameters
uint32_t udpPacketSize = 1000;
// For 2x2 MIMO and NR MCS table 2, packet interval is 40000 ns to
// reach 200 mb/s
Time packetInterval = NanoSeconds(40000);
Time udpAppStartTime = MilliSeconds(400);
```

fig. 3-2. scenario code snippet

In this code snippet we a lot of parameters we will walk into them one by one

apUe.nAntCols = 2

ap. nAntRows = 2

knowing that Ue and the Gnb use a type of antenna called array antenna which support the mimo capabilities

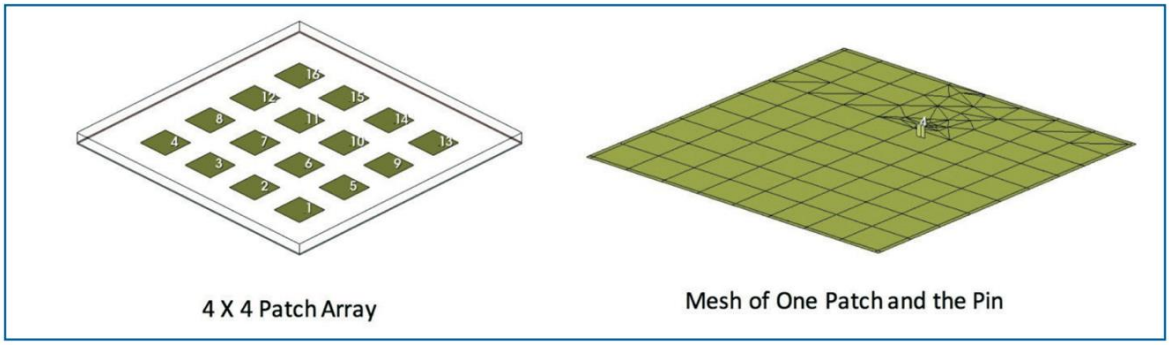


fig. 3-3. diffrence between patch array and one

so, either the antenna of the user equipment or the Gnb is specified by this antenna array which is specified by the number of Rows and the number of columns

the antenna elements is then mapped into specific ports so each number of elements is managed by one port and so for the matter of beamforming refer to the chapter of single user MIMO and beamforming to understand the concept behind it, In SU-MIMO, multiple streams (or layers) are sent to the same UE. Each stream corresponds to a separate port.

The number of streams that the base station can send to the user is determined by a parameter which is learned by the base station unless stated otherwise in the scenario which is the **RANK INDICATOR** which can support to 4 streams per user , Digital precoding maps these streams to ports based on PMI (Precoding Matrix Indicator) feedback from the UE; this mapping is not fixed but dynamically optimized So in short, the UE senses the channel conditions through all the phenomena's that can happened like shadowing or fading and sends the Gnb the index of the precoding matrix that fits this channel conditions

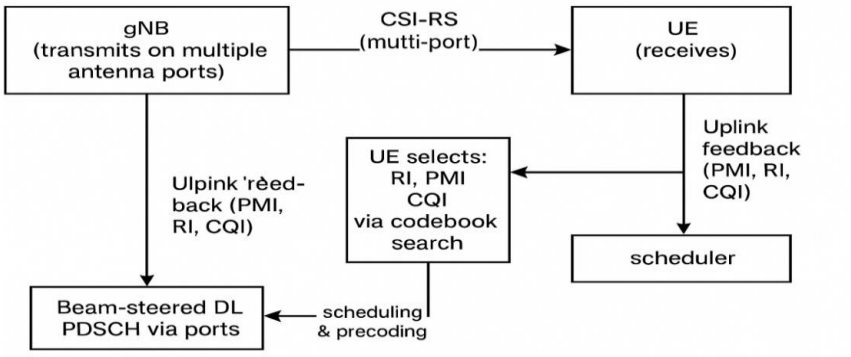


fig. 3-4. how PMI feedback works

apGnb.antennaElem = “ns3::ThreeGppAntennaModel”

There are many types of the antenna that can be used in this scenario if you look in the

antenna models that is available there is like: uniform planar array and the phased array you will find the source files of the antenna models in the src/antenna /model

. isDualpolarized = True; the concept of dual polarization in the antenna theory is as follows

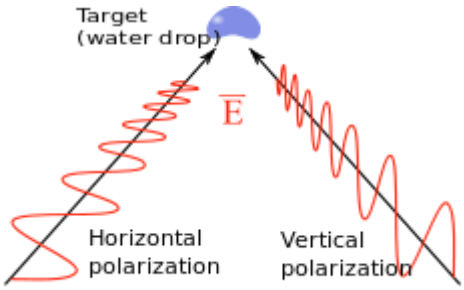


fig. 3-5. dual polarization

In **electromagnetic (EM) theory**, **polarization** refers to the **orientation** of the electric field vector of a radiated wave

Linear polarization: The field vibrates in a single plane (e.g., vertical or horizontal).

Dual polarization: The antenna supports two orthogonal polarization states, typically:

Vertical (V) and Horizontal (H)

Or +45° and -45°

Or Right-hand circular and left-hand circular (in some cases)

So, what is the purpose of the dual polarization Enable two independent data streams on the same frequency and same physical antenna location, but with different polarizations.

Reduces interference and multipath effects.

Improves spectral efficiency without increasing physical antenna size , in this matter it is advised to make the dual polarization parameter in the scenario is True , in the dual polarization antenna each antenna element will have two ports one for the horizontal and one for the vertical one **so if you make the Gnb ports to like 2 ports with dual polarization it will be 4 ports in total** which will improve as stated the spectral efficiency and the ability to send two independent data streams in one frequency which will increase the throughput

These are the default parameters that you can choose to simulate any scenario in the 5G Lena module but in our goal it doesn’t serve the energy saving use case so Eng Mina Yonan our supervisor in this project proposed that we close one of the ports entirely to save the energy that comes from that port , the problem was simple make sure that the power to the port is equivalent to zero Db to make sure that it is consuming no power

3.2 OUR Walkthrough and our tasks

Before diving into any changes, we will walk through all the tasks that we have done.

To help you understand how the change was made and what made us make this specific change

First thing is to know how the NR Module works, we recommend starting with reading the tutorials made by the CTTC (Centre Tecnològic de Telecomunicacions de Catalunya)

Which was responsible for making the NR module
we would like to reference two pdfs to you read to understand the basic principle of the simulation

1. Nr tutorial - > you will find in the reference
2. NR_tutorial - > you will find in the reference

Back on how we can control the power and reduce it the objective was to **decrease the power that the base station is consuming by controlling the power per port in the phy layer of the 5G protocol stack**

First thing what is a port in the context of the NR MODULE

A port can be defined by a logical antenna radiation interface which the baseband is

Transmitted or received corresponding to a distinct beam or polarization

So, what does in detail mean each antenna can be controlled by a port or multiple antenna

Think of the port as the upper supervisor of a group of a certain antenna elements

And it will control the beamforming vectors of each of the antenna elements that is controlled by the process of controlling the antenna elements is called mapping in the context of mobile communications

Mapping between antenna port and physical antenna can be:

One to One

One to Many

One to one mapping is useful when operating in lower frequency bands which do not require beamforming (beamforming requires multiple physical antenna elements). While one to many mappings are useful for beamforming in higher frequency bands.

One to One example, 2*2 MIMO case, antenna ports = 1000 & 1001. Antenna port 1000 is mapped onto one physical antenna while antenna port 1001 is mapped onto another physical antenna

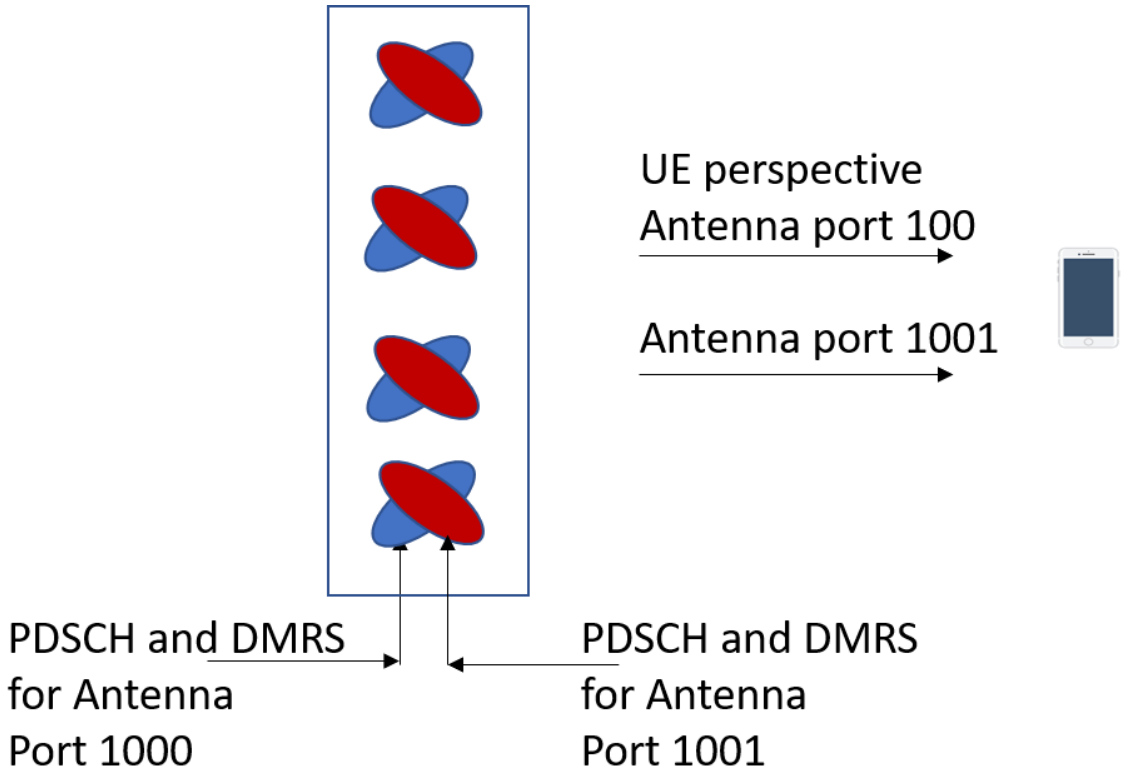


fig. 3-6.port one to one mapping

From the UE perspective, there are two downlink transmissions, one PDSCH and its DMRS associated with antenna port 1000, and another PDSCH and its DMRS associated with antenna port 1001. The UE does not require knowledge of which physical antenna elements were used for the two transmissions.

One to many Mapping

In this case, antenna ports 1000 and 1001 are each mapped onto multiple physical antenna elements. This provides beamforming but with less gain and directivity than the first example because there are fewer physical antenna elements per antenna port

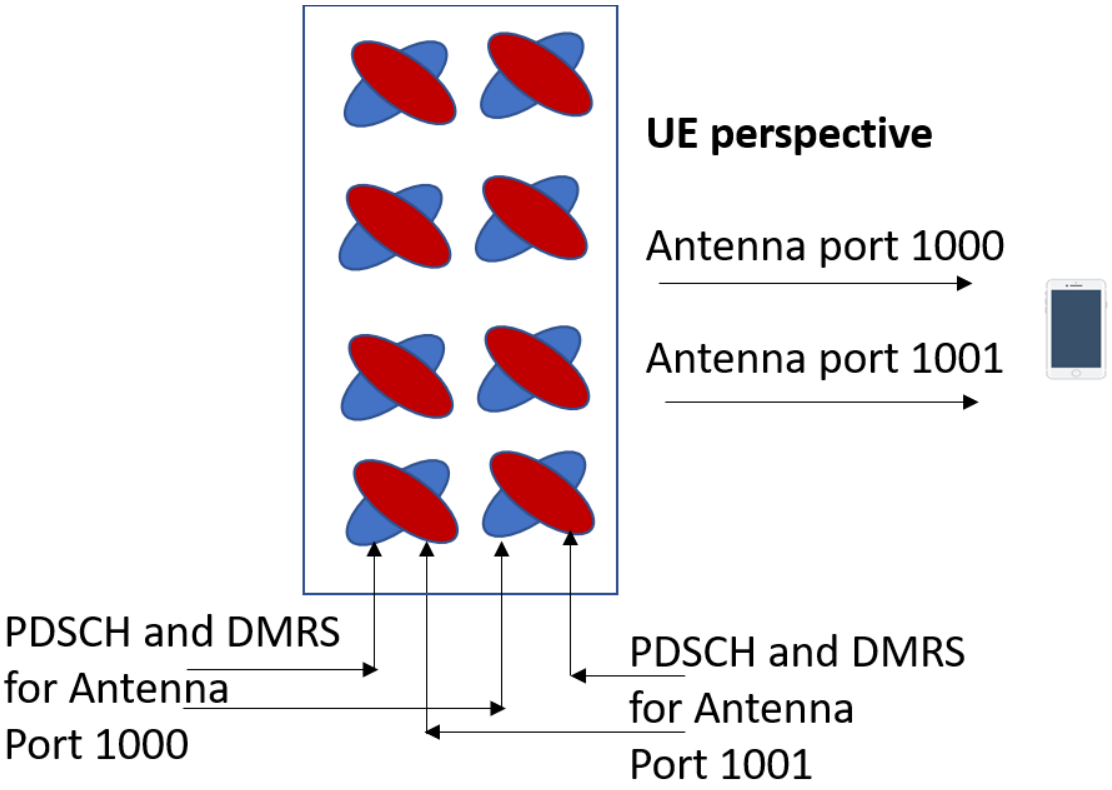


fig. 3-7. ports many to many mapping

- From the UE perspective, the first example and this example are the same, i.e. in both cases, the UE receives antenna ports 1000 and 1001.
- The beamforming helps to improve the link budget but is transparent to the UE, i.e. the UE does not require any explicit knowledge of the beam forming at the Base Station.

That was the concept or a small recap of how the ports works and how we use it one more thing is why do we need ports in the first place can't just all the antenna elements be controlled individually there are three benefits from adding the concept of ports in the antenna

1. Decouples the physical form from the logical

Instead of managing each antenna element individually you can specify a port to control one antenna element or a bunch of them based on the channel conditions and other characteristics

2. Enables codebook-based feedback (CSI Reporting)

In the 5G NR the UE reports the channel state information (CSI) per port not per antenna element so the feedback loop like the PMI (precoding matrix indicator) and the RI (RANK indicator) is defined per port not per antenna So It makes the precoding manageable and enabling efficient use of the feedback

Now as we continue the tasks, we will begin the first task we have been assigned to by ORANGE

First task replicates this graph

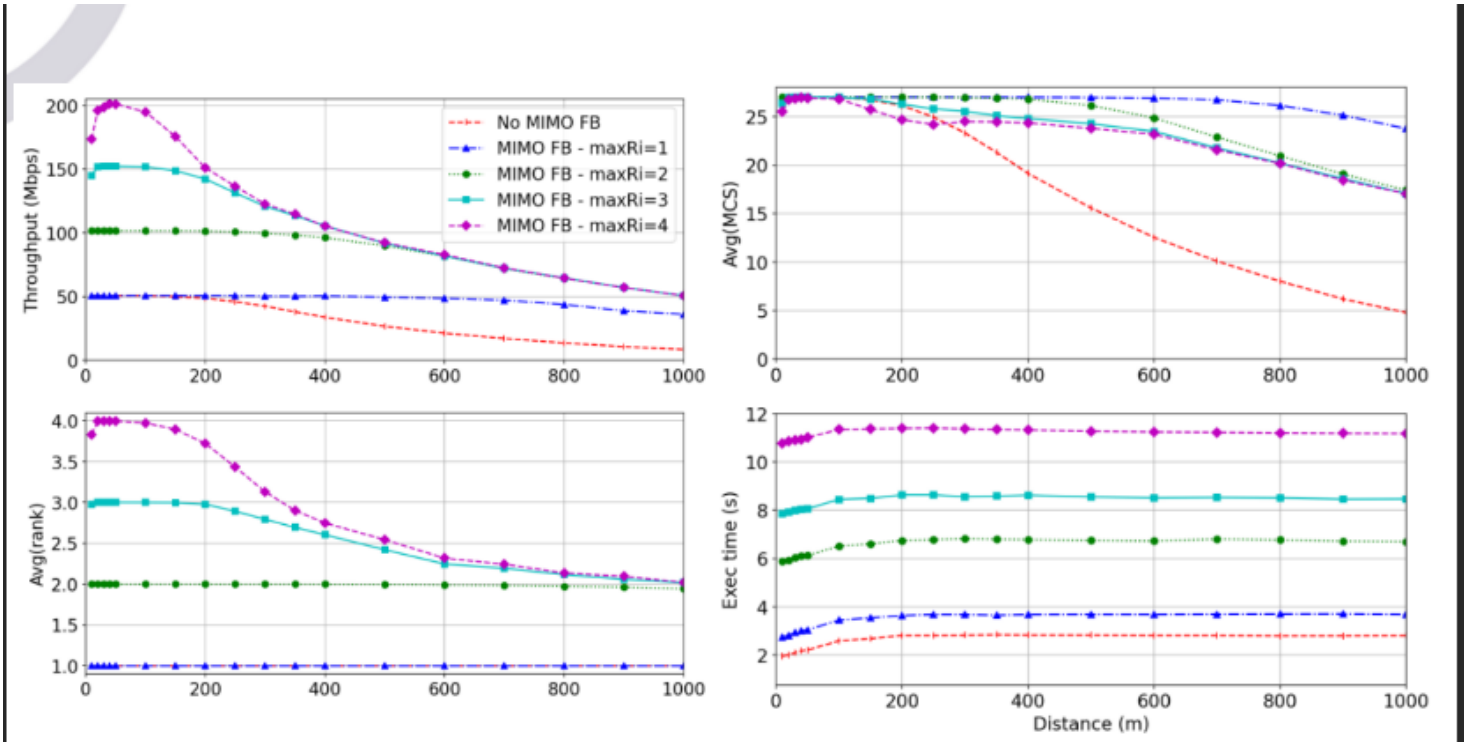


fig. 3-8. cttc-nr-mimo-graphs

This was a relatively easy task. The reason was easy because it was the simulation metrics and changing the distance between the UE and the Gnb when you the default scenario in your terminal, this results will happen

When you run the command ./ns3 run cttc-nr-mimo-demo In the ns3 directory in the terminal This will happen

```
Flow 1 (1.0.0.2:49153 -> 7.0.0.2:1234) proto UDP
  Tx Packets: 6000
  Tx Bytes:   768000
  TxOffered:  10.240000 Mbps
  Rx Bytes:   767744
  Throughput: 10.236587 Mbps
  Mean delay:  0.276044 ms
  Mean jitter:  0.030032 ms
  Rx Packets: 5998
Flow 2 (1.0.0.2:49154 -> 7.0.0.3:1235) proto UDP
  Tx Packets: 6000
  Tx Bytes:   7680000
  TxOffered:  102.400000 Mbps
  Rx Bytes:   7667200
  Throughput: 102.229333 Mbps
  Mean delay:  0.900970 ms
  Mean jitter:  0.119907 ms
  Rx Packets: 5990


  Mean flow throughput: 56.232960
  Mean flow delay: 0.588507
```

fig. 3-9.cttc-nr-mimo-results

Showing the results of two Ue in the scenario

These lines are responsible for creating the number of Gnb and the UEs, so you create and modify the scenario as you like you have worked on a one Gnb and one Ue

```
1
2   NodeContainer gnbContainer;
3   gnbContainer.Create(pairsToCreate);
4   NodeContainer ueContainer;
5   ueContainer.Create(pairsToCreate);
```

fig. 3-10.creating Gnb, Ue

So, by doing this we can create the results for a specific scenario by running the script for one time

So, by running the script multiple times and capturing the values of the throughput and the jitter and the delay over the multiple distance and storing them we can graph this

the solution to this task we to use **shell scripting** and use **a bash file** with the with

The scenario as the base command and passing the parameters that we needed in the terminal and storing the values of throughput and jitter and delay in a csv

```bash
#!/usr/bin/bash

# Define the base command for running the ns-3 simulation
base_command="./ns3 run scratch/trail_error --"

# Define the CSV file for the simulations
csv_file="norm_off_port02502500.csv"

# Write the header if the file does not exist
if [ ! -f "$csv_file" ]; then
    echo "portPowerLabel,enableMimoFeedback,rankLimit,gnbUeDistance,Metric,Value" > "$csv_file"
fi

# Set fixed values as per requirements
enable_mimo_feedback=1  # MIMO feedback enabled by default
rank_limit=4            # Rank fixed to 4

# Loop through distances from 0 to 1000 with an increment of 20
for distance in $(seq 0 20 1000); do
    # Construct the full simulation command with default parameters
    full_command="$base_command --numColumnsGnb=2 --numRowsGnb=2 --numHPortsGnb=2 --numVPortsGnb=1 --xPolGnb=1 --numColumnsUe=2
```

fig. 3-11.bash script for graphs

This is a part of the bash file it runs the base command and stores the value in a csv file of your choice in this scenario make sure we have activated the mimo feedback which reports the PMI to the Gnb again for maximum SINR

You should run this bash file inside the ns3 directory, or it won't work

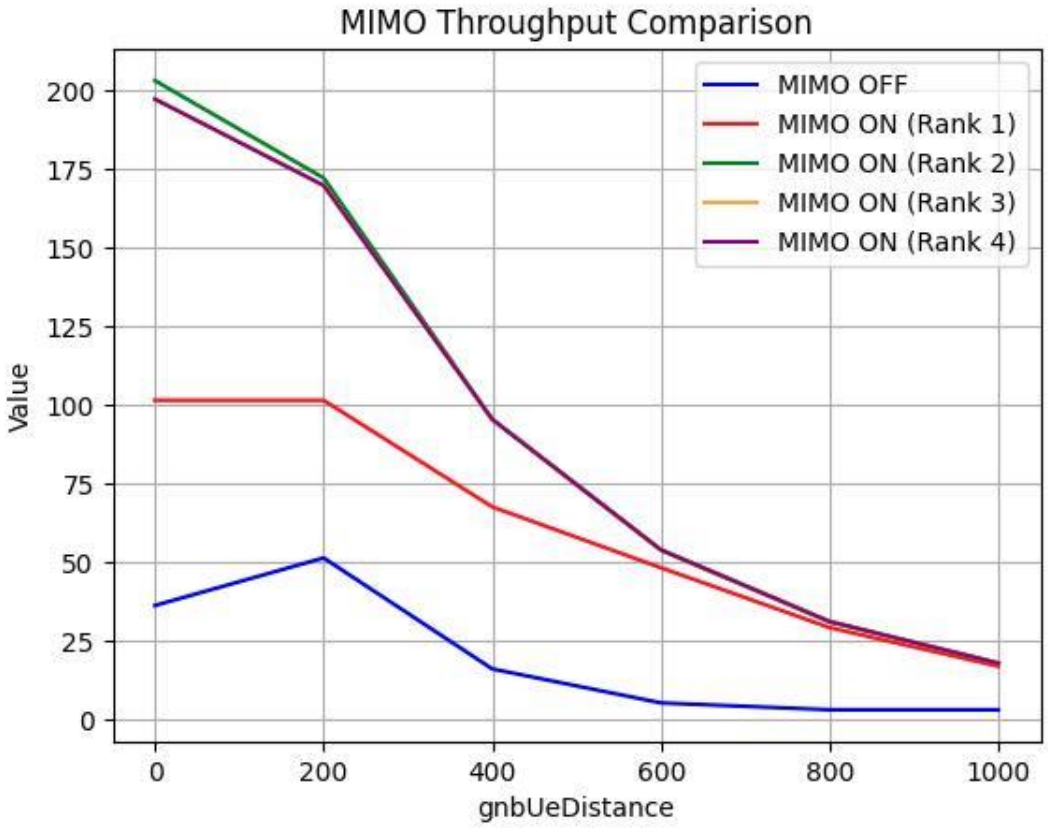


fig. 3-12. MIMO throughput graph

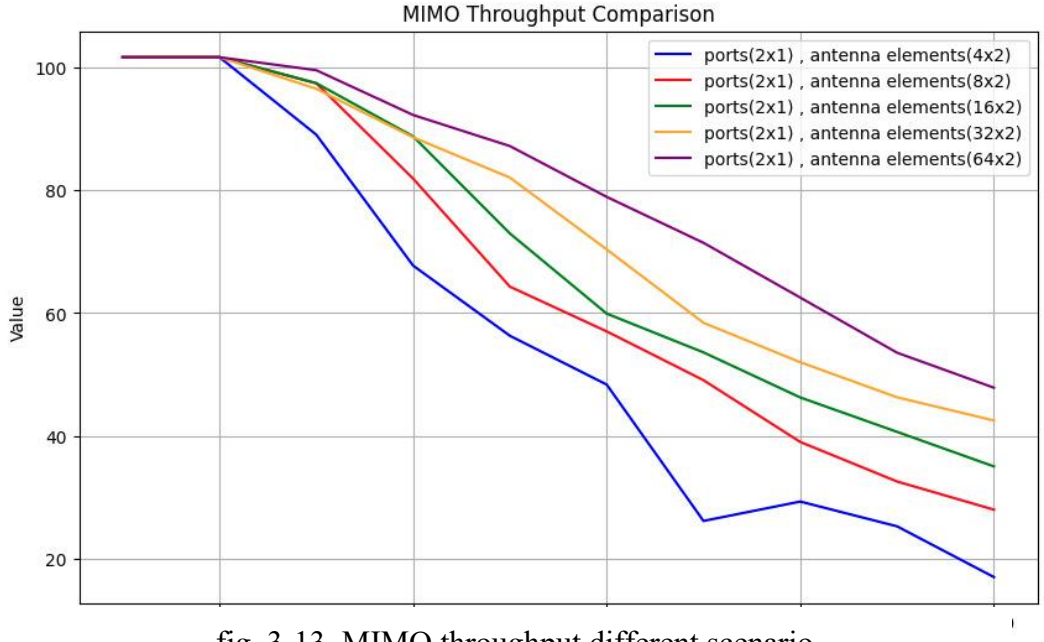


fig. 3-13. MIMO throughput different scenario

Some insights from both graphs

1. The obvious one is the throughput decrease as we increase the distance
2. The higher the RANK INDICATOR RI the higher the throughput in the further distances

   As the number of rank increases the number of streams that is sent to the ue
3. The third one is that increasing the number of antenna elements the

   The higher the throughput the higher distances but this comes with a cost all these antenna elements are consuming very high power at the cost of the higher throughput

Second Task: reducing the power Via closing the port of the Gnb

This task was the hardest one and most time consuming the reason for that is the neither the Ns3 nor the 5G Lena support the power per port

how the 5G Lena distribute the power to the antenna

The NR module supports two types/models for power allocation:

• uniform power allocation over the whole BW, i.e., power per RB is

fixed

• uniform power allocation over the active RBs, i.e., power per RB

depends on the number of active RBs

• If an RB is not allocated to any data transmission, the transmitted

power over it is 0, and no interference is generated in that RB

• Implemented in: nr-spectrum-value-helper.h/cc

• Configurable through the attributePowerAllocationType of NrGnbPhy

and NrUePhy class

in the default source files there was no sign in the ability to control the power per port

the power was distributed over the whole bandwidth of the transmission

The transmission is divided into multiple frequencies channels, and our goal was to

control the power per port to introduce the concept of energy saving and then make the use case into a scenario we can Implement in the NS3 and the 5G Lena Module

```
Ptr<SpectrumValue>
NrSpectrumValueHelper::CreateTxPsdOverAllRbs(double powerTx,
                                             const std::vector<int>& activeRbs,
                                             const Ptr<const SpectrumModel>& spectrumModel)
{
    NS_LOG_FUNCTION(powerTx << activeRbs << spectrumModel);
    Ptr<SpectrumValue> txPsd = Create<SpectrumValue>(spectrumModel);
    double powerTxW = std::pow(10., (powerTx - 30) / 10);
    double txPowerDensity = 0;
    double subbandWidth = (spectrumModel->Begin()->fh - spectrumModel->Begin()->fl);
    NS_ABORT_MSG_IF(subbandWidth < 180000,
                    "Erroneous spectrum model. RB width should be equal or greater than 180KHz");
    txPowerDensity = powerTxW / (subbandWidth * spectrumModel->GetNumBands());
    for (int rbId : activeRbs)
    {
        (*txPsd)[rbId] = txPowerDensity;
    }
    NS_LOG_LOGIC(*txPsd);
    return txPsd;
}
```

fig. 3-14. how 5G Lena allocate power

This function is responsible for the distribution of power across the active Rbs (resource blocks) it returns a spectrum value object which contains the power density for each resource block over the frequency spectrum. It takes the power TX by Db and converts it into linear scales and divides it over the frequency spectrum

This is one of the methods of power allocation as mentioned above

so we had to find a way to make a function to be able to control the power distribution around the ports of the system

the next section we will explain what the function is and what is the thought process

3.3 our contribution to Ns3 and 5G Lena

In the figure above the scenario parameters, you might notice a line that sounds Interesting which is

```
apGnb.port_power = {1.0, 1.0, 1.0, 1.0 }; // port power for each antenna element
```

fig. 3-15.port power parameter

In the paper "MIMO in network simulators: Design, implementation and evaluation of single-user MIMO in ns-3 5G-LENA" you can follow the link or the reference number 5

In this paper in the section of 3.6

It talks about the implementation of the precoding matrix and how it can affect the beam-forming vectors of the antenna

First thing, what is the precoding matrix?

Is a matrix helping of the mapping of the antenna the streams to the antenna ports as it controls the beamforming vectors and steering of the antenna like phased array

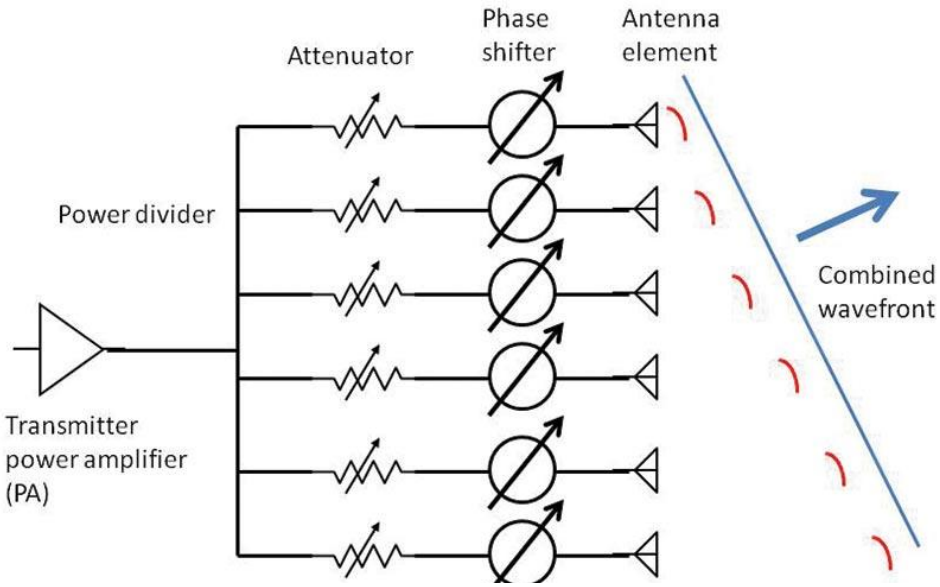


fig. 3-16.how group of antennas steer a beam

Sounds complicated Right? **Let's break it down to you**

The precoding matrix is like a way to tell the ports which control a set of antenna elements

How to steer the beams towards the user or to apply different channel conditions based on the scenario you are in the precoding matrix is a normal matrix but the value inside of it

Are complex values like this 3+5i

The real value tells the value of the amplitude of the beam while the complex value tells the

Angle of steering in radian

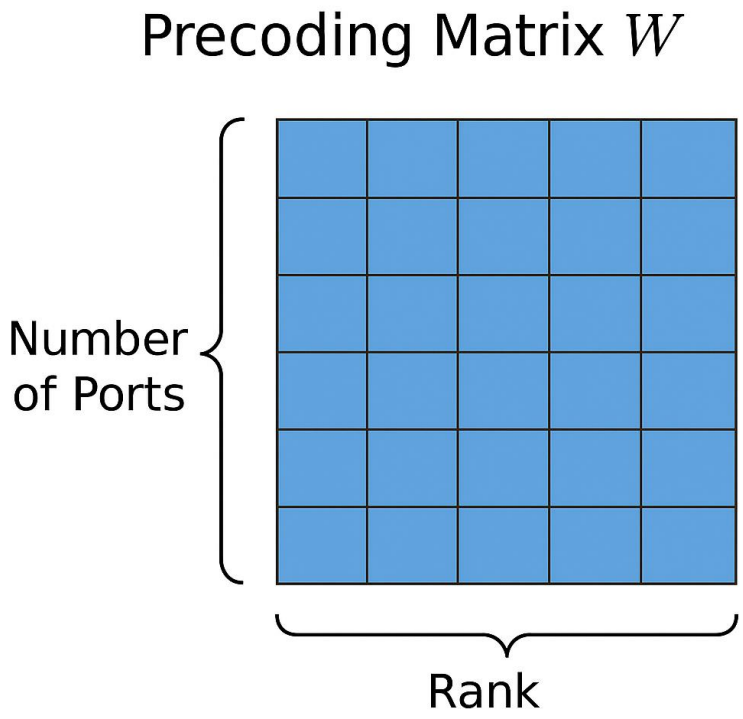


fig. 3-17.precoding matrix

We have explained the ports above how the rank influences the number of streams that the antenna can send

When you look at an instance for example W (1,1) and let's make its equal to 1+0j that means I will give that port that controls a set of antenna elements a power of 1

The power of 1 can be anything like 30Dbm or 60Dbm or whatever you choose

That means the beam number 1 will be directed by port 1 in the angle of zero degrees

And the same for the other ports and ranks

So, this is the basic working principle

By this observation we observed why don't we control how much power that runs through each port in this precoding matrix by introducing a new parameter called port power. The main goal of port power is to control the power of each port so you can shut down the power of specific antenna elements

we went to modify the file of the precoding matrix which is the nr-cp-type-one-sp.cc

which is new radio – codebook – type -one – single panel

so, this is the code which we modified

```cpp
NrCbTypeOneSp::GetBasePrecMatFromIndex(size_t i11, size_t i12, size_t i13, size_t i2) const
{
    if (m_nPorts == 1)
    {
        auto res = ComplexMatrixArray(1, 1);
        res(0, 0) = 1.0;
        return res;
    }

    // m_nPorts is even-numbered. The upper half of ports represent the first polarization angle
    auto precMat = ComplexMatrixArray(m_nPorts, m_rank);
    auto phase = M_PI * static_cast<double>(i2) / 2.0;
    auto phiN = std::complex<double>{cos(phase), sin(phase)}; // phi_n as defined in 5.2.2.2.1
    auto normalizer = 1.0 / sqrt(m_nPorts * m_rank);
    auto uniqueBfvs = CreateUniqueBfvs(i11, i12, i13);
    // Quick inline solution
//     std::cout << "uniqueBfvs contents:\n";
//     for (size_t i = 0; i < uniqueBfvs.size(); ++i) {
//         for (size_t j = 0; j < uniqueBfvs[i].size(); ++j) {
//             std::cout << "(" << uniqueBfvs[i][j].real() << "," << uniqueBfvs[i][j].imag() << ") ";
//         }
//     std::cout << "\n";
// }
    for (size_t layer = 0; layer < m_rank; layer++)
    {
        // The beamforming vector for the first polarization
        auto v = uniqueBfvs[m_uniqueBfvInds[layer]];

        NS_ASSERT_MSG(v.size() == m_nPorts / 2,
                      "Size of a per-polarization beamforming vector must be nPorts/2");
        for (size_t vIdx = 0; vIdx < v.size(); vIdx++)
        {
            // Fill in the precoding matrix W for both the first and second polarization
            // Apply port power scaling if power allocation is active
            if (IsPowerAllocationActive())
            {
                // First polarization
                precMat(vIdx, layer) =  normalizer *   v[vIdx]*(m_portpower[vIdx]);
                // Second polarization
                precMat(vIdx + v.size(), layer) =   normalizer *   m_signPhiN[layer] * phiN * v[vIdx]
                                                  * (m_portpower[vIdx + v.size()]);
            }
            else
            {
                // Use original values without port power scaling
                precMat(vIdx, layer) = normalizer *  v[vIdx];
                precMat(vIdx + v.size(), layer) = normalizer * m_signPhiN[layer] * phiN * v[vIdx];
            }
        }
```

fig. 3-18.precoding matrix code

first thing is the three parameters

i11 - Wideband/Horizontal Beam Selection:

Selects the primary horizontal beam direction

Creates DFT vectors that point the beam in specific azimuth angles

Typically ranges from 0 to O1-1 where O1 is the horizontal oversampling factor

Formula: Creates vectors like exp (j * 2π * i11 * n / N1) where N1 is horizontal antenna count

i12 - Wideband/Vertical Beam Selection:

Selects the primary vertical beam direction

Creates DFT vectors for elevation beam steering

Ranges from 0 to O2-1 where O2 is the vertical oversampling factor

Formula: Creates vectors like exp (j * 2π * i12 * m / N2) where N2 is vertical antenna count

i13 - Sub band Beam Refinement:

Provides frequency-selective beam adjustment

Allows fine-tuning of the beam per sub band

Enables beam squint correction at different frequencies

Creates additional phase shifts for frequency-dependent optimization

These three parameters help to make the beamforming vectors so they can adapt to the scenario

If the port is just one, it will just create a instance and its value will be 1+0j

Where there is only one beam, make sure the value of the rank can't be higher than the value of the ports in the first place and that is the if statement

If not, there are some steps that it takes to manage it first it creates a complex matrix

Of the size of num_of_ports X Rank to store the value of the beamforming vectors in the matrix

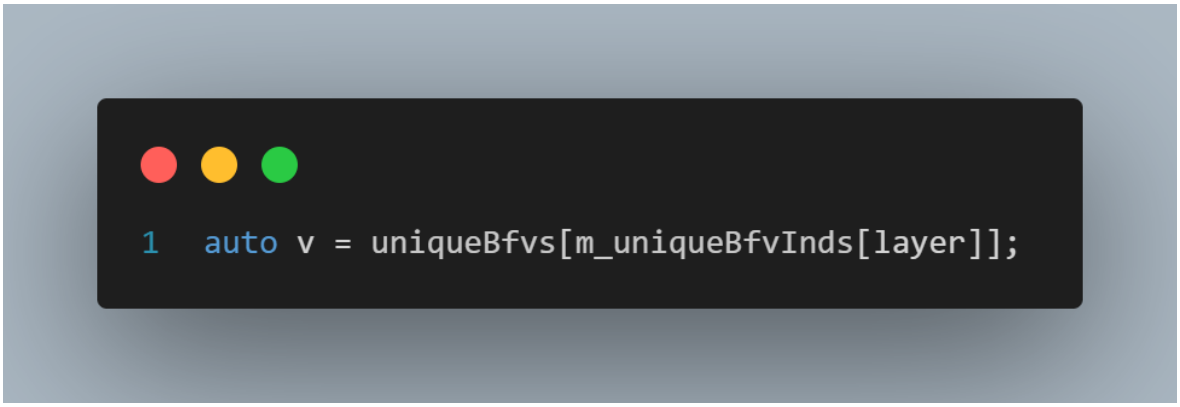
```
auto phase = M_PI * static_cast<double>(i2) / 2.0;
auto phiN = std::complex<double>{cos(phase), sin(phase)}; // phi_n as defined in 5.2.2.2.1
```

fig. 3-19.phase difference between polarizations

in this code snippet it just calculates the phase difference between the first and the second

polarization the first half of the precoding matrix is concerned of the first polarization while the second half is concerned with the second half **in** this matter a new phase shift should be between the first and second polarization as stated in the previous section

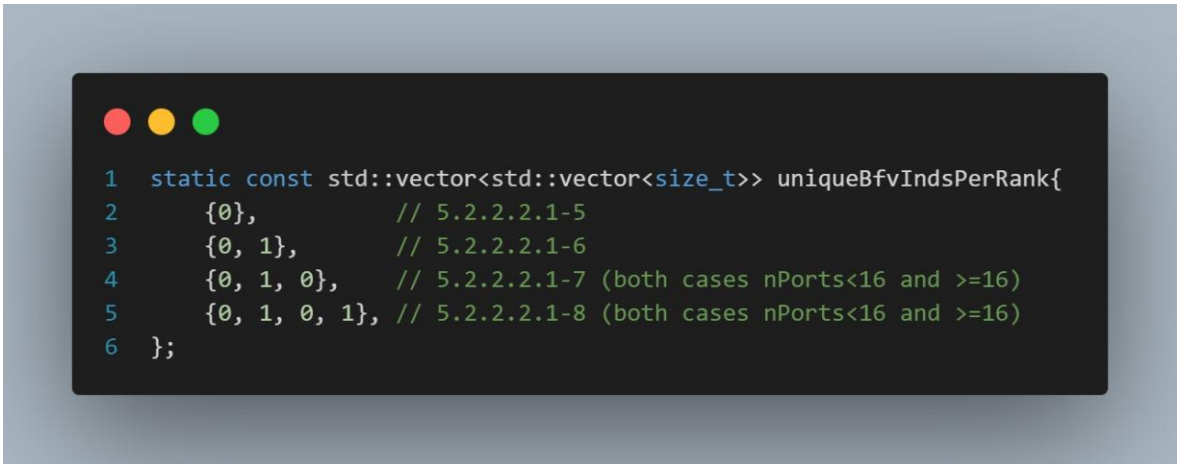
After that it loops through all the layers. The layers here mean the Rank indica



```
1  auto v = uniqueBfvs[m_uniqueBfvInds[layer]];
```

fig. 3-20.creating beamforming vectors

In this code it chooses the beamforming vector based on the rank, to minimize the interference between the layers certain beamforming vectors must be used to not let this happen so for each layer a certain beamforming vectors is used so how does this happen?



```
1  static const std::vector<std::vector<size_t>> uniqueBfvIndsPerRank{
2      {0},          // 5.2.2.2.1-5
3      {0, 1},       // 5.2.2.2.1-6
4      {0, 1, 0},    // 5.2.2.2.1-7 (both cases nPorts<16 and >=16)
5      {0, 1, 0, 1}, // 5.2.2.2.1-8 (both cases nPorts<16 and >=16)
6  };
```

fig. 3-21.beamforming per Rank

This code based on the Rank which is maximum 4 by the way it can specify the index of the beamforming vector is chosen at each rank in this example

This auto v = uniqueBfvs[m_uniqueBfvInds[layer]];

Gives you two beamforming vectors at index 0 , 1 when make the Rank indicator = 4 the first layer = beamforming_vector(0)

The second layer = beamforming_vector (1)
the third layer = beamforming_vector(0)

The fourth layer = beamforming_vector (1)

And so in this way you can minimize the interference between the layers and each other

After that you must fill these values into the matrix so you loop first in the first half of the matrix knowing that the other half the second half will be the same values but with a phase shift as stated above

```
if (IsPowerAllocationActive())
{
    // First polarization
    precMat(vIdx, layer) =  normalizer *   v[vIdx]*(m_portpower[vIdx]);
    // Second polarization
    precMat(vIdx + v.size(), layer) =   normalizer *   m_signPhiN[layer] * phiN * v[vIdx]
                                      * (m_portpower[vIdx + v.size()]);
}
else
{
    // Use original values without port power scaling
    precMat(vIdx, layer) = normalizer * v[vIdx];
    precMat(vIdx + v.size(), layer) = normalizer * m_signPhiN[layer] * phiN * v[vIdx];
}
}
```

fig. 3-22. power allocation code

And here comes the magic part

You loop through the ports and fill them with the precoding matrix but with a catch

Remember the parameter called port power in the scenario parameters this passed here into the source file of the codebook to control the power of each port

the mathematics here is to put the amplitude of each port we want into zero that means

No power will come from this port even if there is a phase shift

fig. 3-23. port power allocation

In this example I am giving full power into all the ports in another example we can set the **first two numbers with zeros with that you can disable 50% of the antenna power**

With just changing the amplitude of the port in the precoding matrix , this change alone helped us achieve the concept of decreasing the power of the whole system using only the port control , without shutting down the base station for an option because it is an expensive option and the restarting of the base station can take up to 15 minutes which is long time

First, we conducted the throughput vs distance graph to see if the change is working

But before that how can you run the scenario to see the results for yourself

1. make the change of the antenna port power in the scenario
2. Save it
3. Go into the terminal and navigate into the ns3 dev directory
4. Type ./ns3 build , the build will continue
5. In the final step run chmod +x port_simulation.sh to make the bash file executable
6. Type ./port_simulation

And remember to make sure you are in the ns3 repository

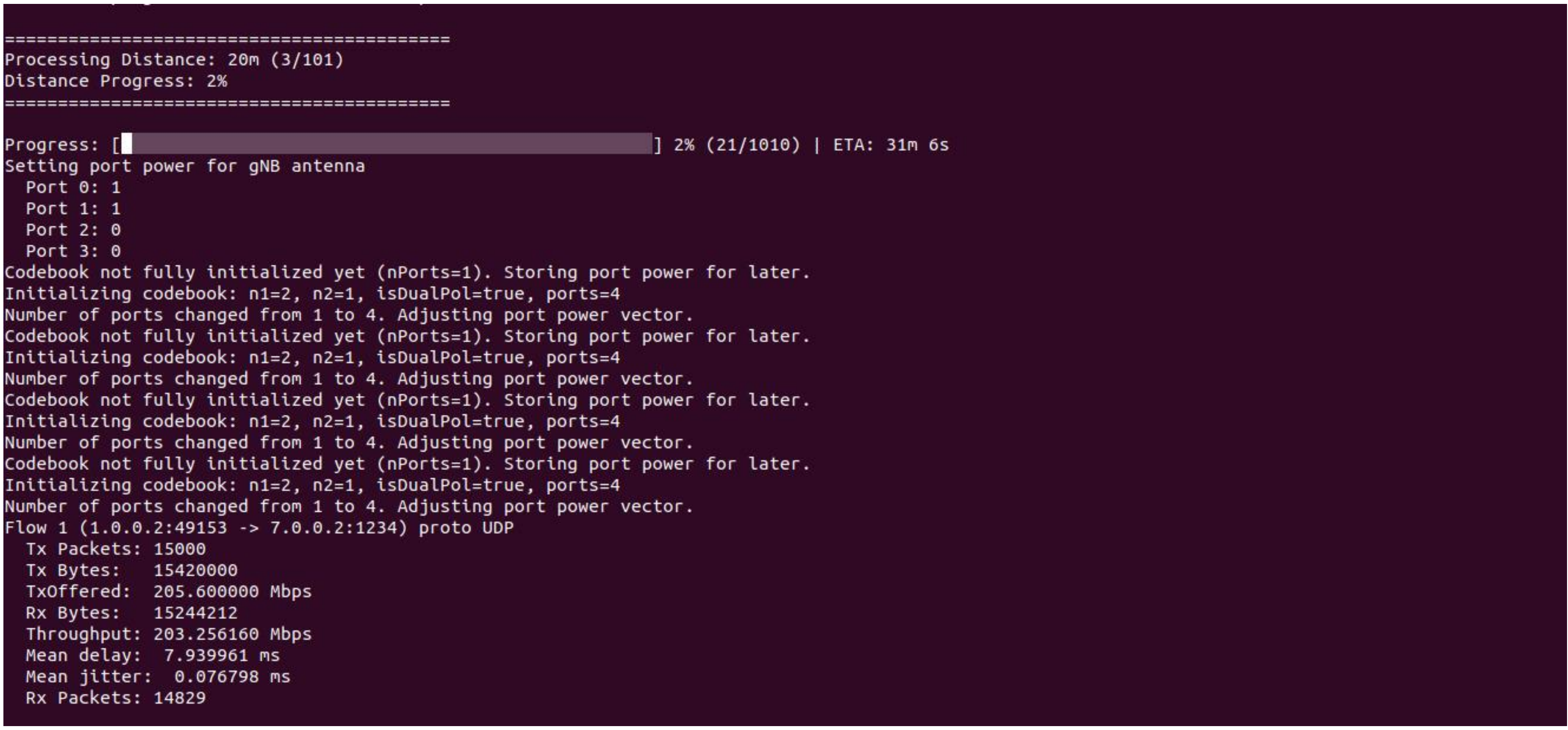


fig. 3-24.results in terminal

You will see something like this telling you the distance covered and the power at each port with the value that you put inside the scenario after that you go to a python file
Jupiter notebook and graph the csv file of your choice

Results

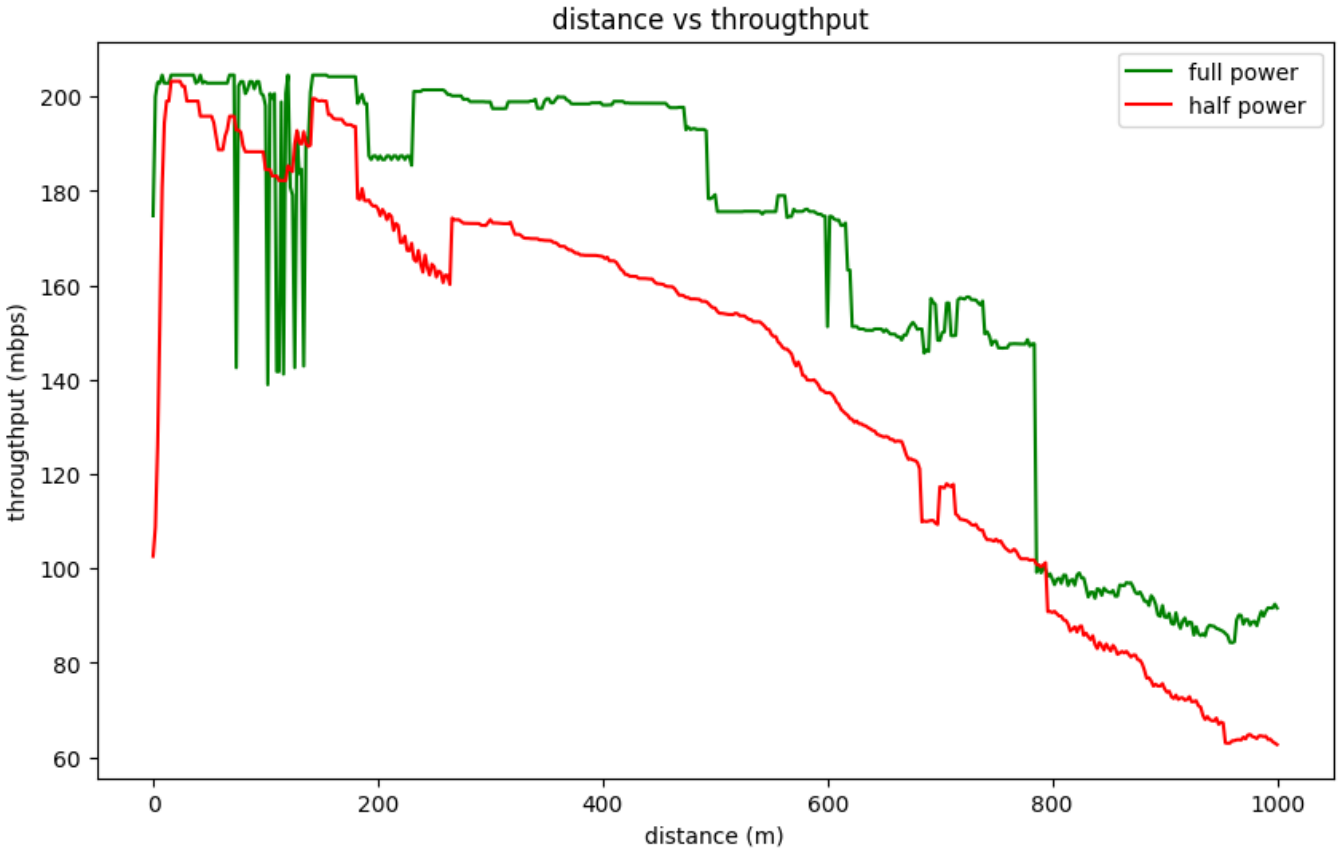


fig. 3-25. half_power vs full power

You will notice the normal simulation behavior when the port power is cut to half aka

You shut down two out of the four ports the throughput behavior the throughput behavior

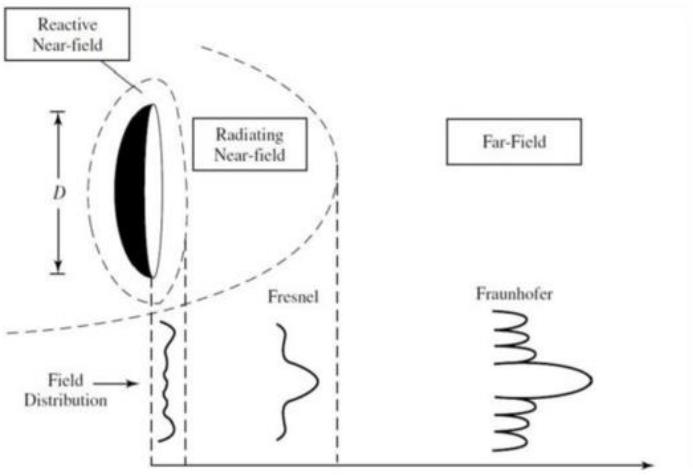


fig. 43. far field region

The throughput is very low at higher distances like 1000 meters you will notice dip at the distance of the 100m range this is the because the far field region is destructive in this area

Will find the expected behaviors also in the delay and the jitter graph

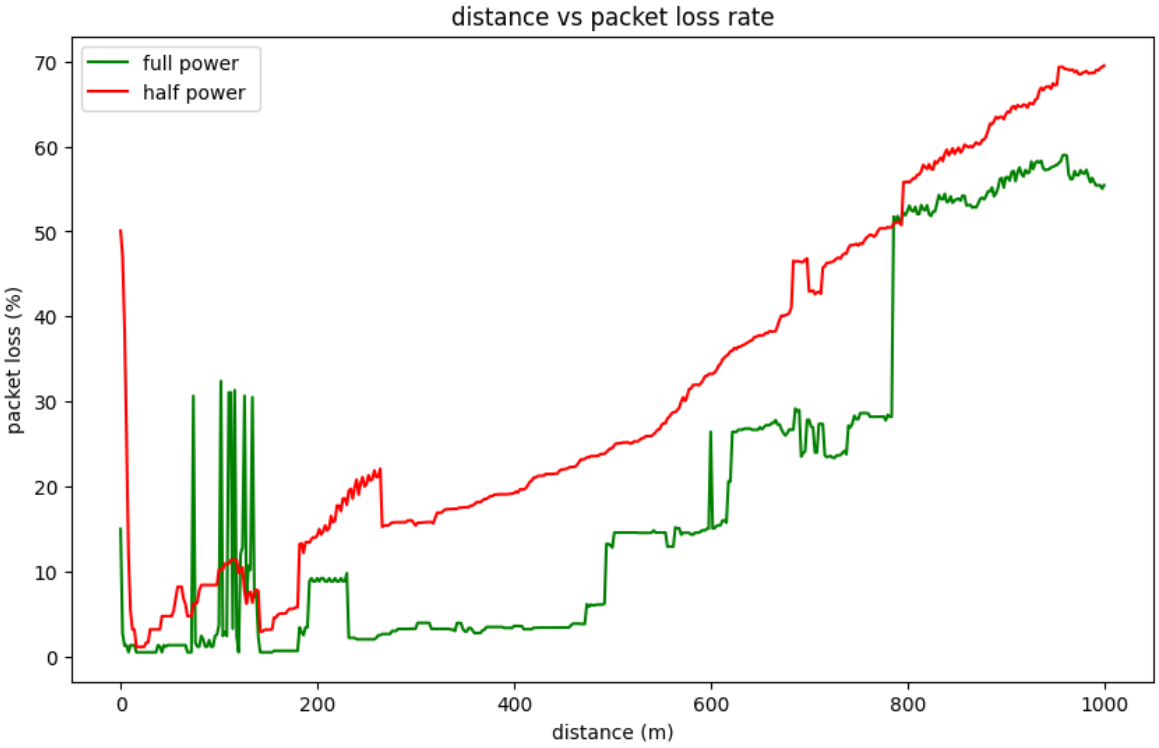


fig. 3-26. packet loss vs distance

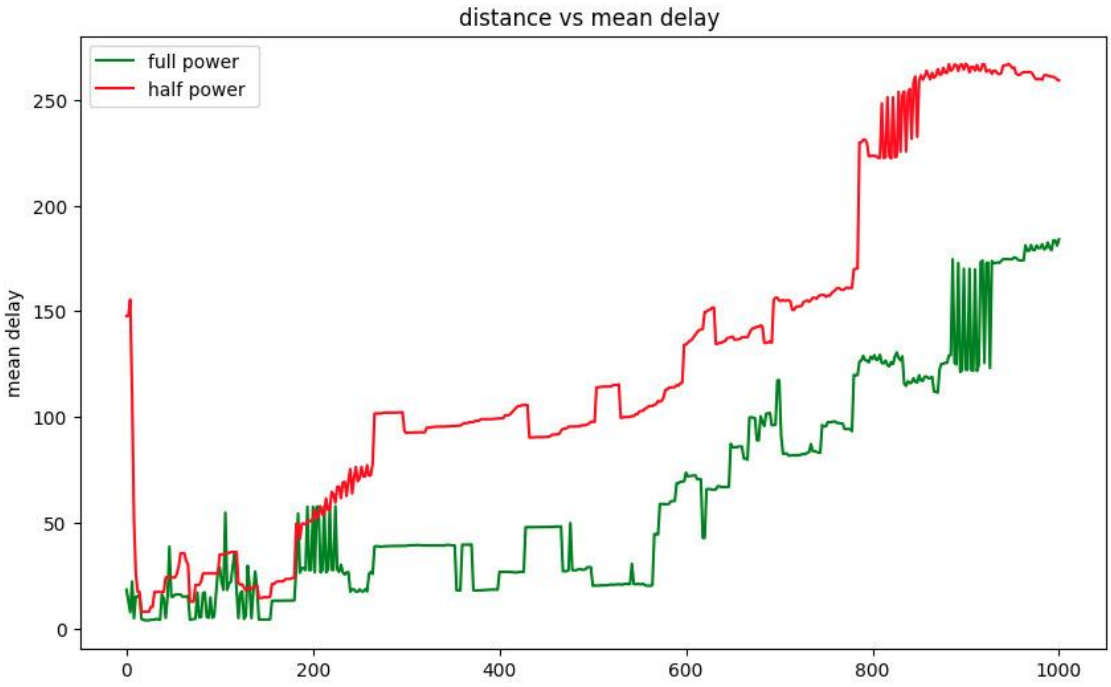


fig. 3-27. distance vs delay

The performance metrics on the half power was a little worse than the full power but this was the whole point of the scenario is to save power in the cost of some metrics

Now to the fun part how much power did we save on the cost of the half port power the change was linear change when we closed half of the ports the power decreased about 50 percent! the change was huge, and it will save the operators which will adapt the

Scenario a huge saving in the OPEX costs

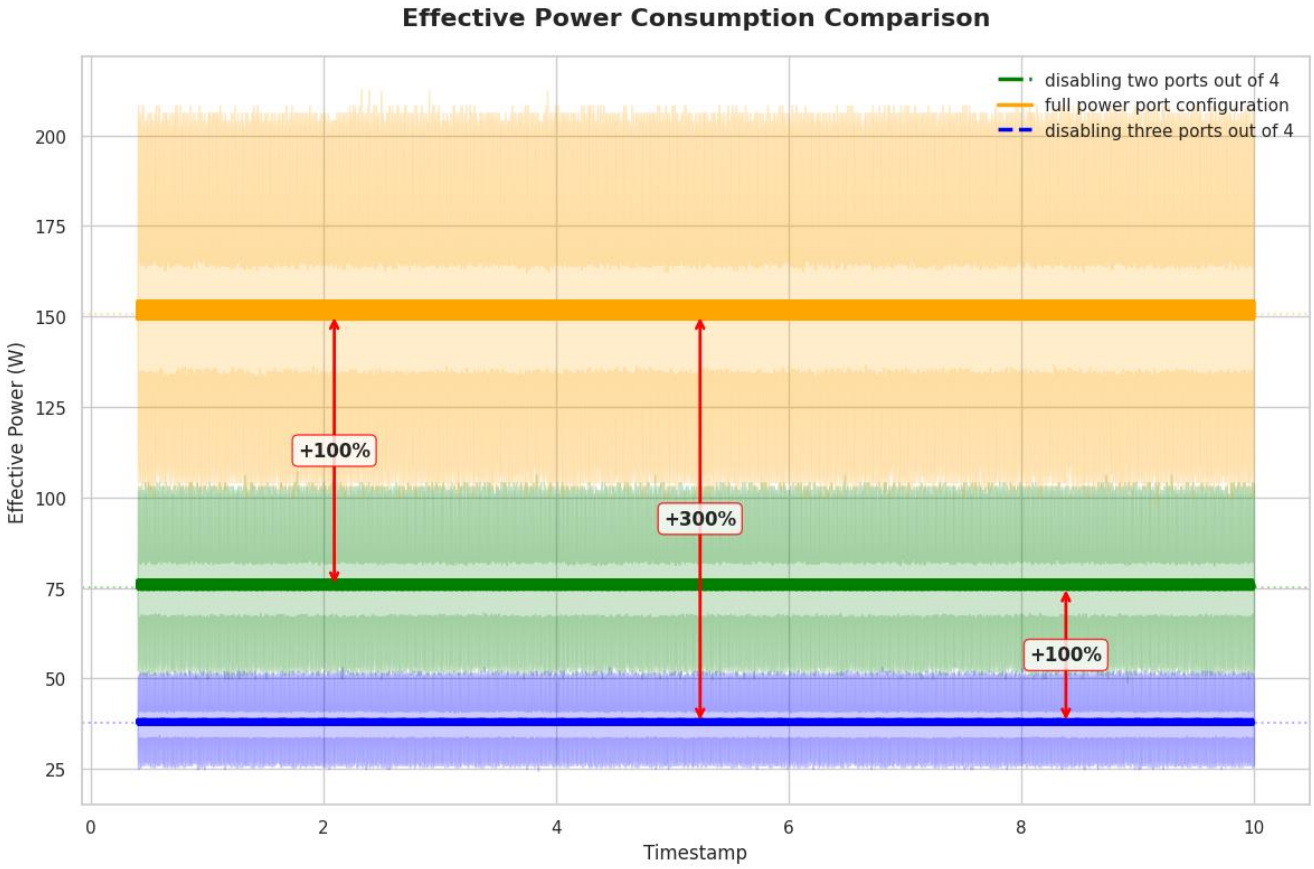


fig. 3-28.power calculation

In this case the shutdown of two ports out of 4 ports will save the operator 50 percent of the power as said and closing three ports will result a change in 75 percent of power decrease

This if you suppose we you close the base station and only use half the power all day but this won't happen

For example, let’s say

You will apply this from 12 AM to 8 AM every day in the year and the price of the KWh in the commercial use here in Egypt is about 2.34egp / kwh

And the average power consumption of the base station per day is 340kwh

The average power consumption per year = 340 kwh * 365 days = 124100 kwh

And the average price of the kwh is about 2.34 egp

The total cost in the year is 2.34 egp * 124100 = 290,394 egp

With the half power scenario of shutting down two ports in the non-peak hours

Table 1. power reduction calculation

Metric	Amount
Daily Energy Saved	340 kWh – 283.3 kWh = 56.7 kWh
Daily Cost Saved	795.6 EGP – 663.0 EGP = 132.6 EGP
Annual Energy Saved	56.7 kWh × 365 = 20 700 kWh
Annual Cost Saved	132.6 EGP × 365 = 48 400 EGP
Percentage of Energy Saved	(56.7 ÷ 340) × 100% ≈ 16.7%

The operator will save about 17 percent of the total power cost and will save about 50000EGP in just one base station with just one change of scenario

And this was the end of the tasks of part 1

4. chapter 4 Energy saving use case using NON-RT RIC

4.1 RAN-PM

4.1.1 introduction

Building upon the foundational exploration of energy efficiency and RF channel reconfiguration in the first part of this project, this section delves into the advanced implementation of Radio Access Network Performance Management (RANPM) within the O-RAN (Open Radio Access Network) architecture. The rapid evolution of wireless communication systems, particularly with the deployment of 5G and the anticipation of future generations, necessitates robust mechanisms for performance monitoring and optimization. RANPM, as a critical component of the O-RAN ecosystem, addresses this need by enabling the collection, processing, and analysis of performance data to enhance network efficiency and reliability.

This chapter focuses on the practical deployment of RANPM within a Kubernetes-based environment, as detailed in the O-RAN Software Community (O-RAN SC) documentation. It provides a comprehensive guide to setting up RANPM, configuring performance logging jobs, and integrating it with the Artificial Intelligence and Machine Learning Framework (AIMLFW) to enable data-driven optimization. By leveraging open interfaces and virtualized components, RANPM facilitates intelligent control through the Non-Real-Time RAN Intelligent Controller (Non-RT RIC), supporting energy-saving strategies and advanced analytics. This work aims to bridge theoretical concepts with practical applications, offering insights into the deployment of open-source tools for next-generation wireless networks and contributing to the broader adoption of intelligent, energy-efficient RAN solutions.

Overview

The OSC Non-RT RIC provides a high performing, fully scalable end-to-end solution for handling PM Measurements. A PM report (containing aggregated PM measurements over a time interval) is an XML file. The format is defined by 3GPP (TS 32.432 and 3GPP TS 32.435). The files are collected from the RAN and stored. An Rapp can subscribe for chosen measurement types from measured resources in the network

4.1.2 components

The picture illustrates the components involved.

- The PM Data File Collector fetches the XML files from the RAN.
- The PM File Converter converts these to a Json format. The structure and the contents are the same as the XML format.
- The PM Producer handles filtering and distribution of PM data to subscribers. These subscribers can be Rapps.
- The Influx Logger stores selected PM measurements into a time series database.
- HTTPS-SERVER is for testing and implements functionality to simulate file transfer from RAN nodes.

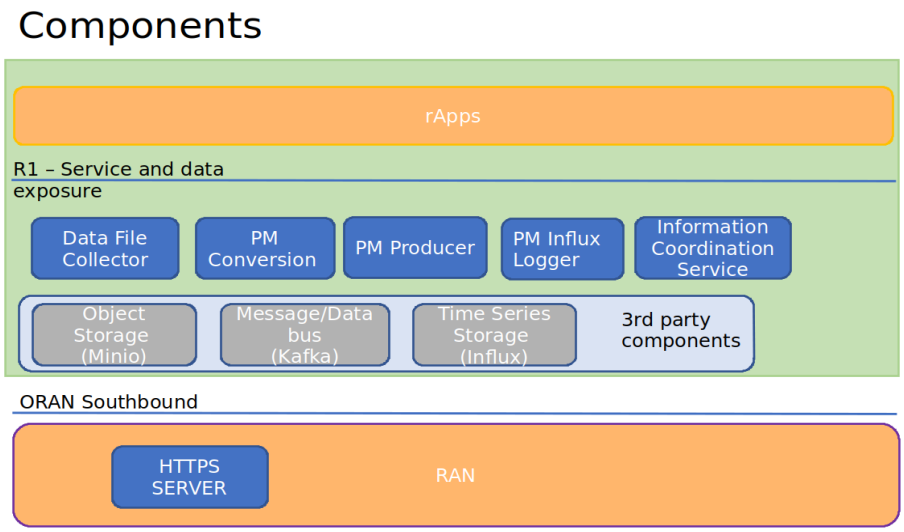


fig. 4-1. Ran pm components

The third-party products used are:

- Minio object storage, for storing of files.
- Kafka for transferring data (not the full PM reports, though)
- Influx time series database for storing selected PM measurements over time.

Non-RT RIC PM Data File Collector

The task of the Data File Collector is to collect OAM data files from RAN traffic-handling nodes. The main use case is (see also the picture below)):

- The DFC receives a “File Ready” VES event from a Kafka topic. This contains a list of all available files.
- The DFC fetches files that are not already fetched from the relevant RAN traffic-handling nodes. This is done using one of the supported file transfer protocols.
- Each file is stored in an S3 Object Store bucket or in the file system (in a persistent volume).
- For each stored file, a “File Publish” message is sent to a Kafka topic for further processing.

The “File Publish” message can be subscribed by other components, which can then read the fetched file and process it further

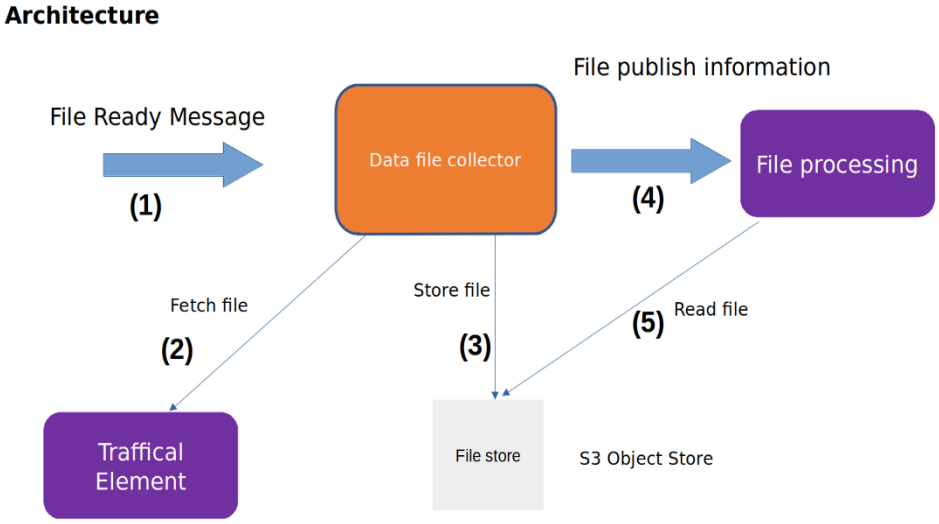


fig. 4-2.overview of file processing architecture

Supported file transfer protocols are:

SFTP
FTPES
HTTP
HTTPS

The service is implemented in Java Spring Boot.

This product is a part of NONRTRIC.

Input File Ready VES Event

Here follows an example of the expected input object:


```
{
 "event":{
    "commonEventHeader":{
       "sequence":0,
       "eventName":"Noti_RnNode-Ericsson_FileReady",
       "sourceName":"5GRAN_DU",
       "lastEpochMicrosec":151983,
       "startEpochMicrosec":15198378,
       "timeZoneOffset":"UTC+05:00",
       "changeIdentifier":"PM_MEAS_FILES"
    },
    "notificationFields":{
       "notificationFieldsVersion":"notificationFieldsVersion",
       "changeType":"FileReady",
       "changeIdentifier":"PM_MEAS_FILES",
       "arrayOfNamedHashMap":[
          {
             "name":"A20220418.1900-1915_seliitdus00487.xml",
             "hashMap":{
                "fileFormatType":"org.3GPP.32.435#measCollec",
                "location":"https://gnb1.myran.org/pmfiles/",
                "fileFormatVersion":"V10",
                "compression":"gzip"
             }
          }
       ]
    }
 }
}
```


fig. 4-3.input file ready VES event

Output File Publish Message

```
{
  "productName":"RnNode",
  "vendorName":"Ericsson",
  "lastEpochMicrosec":151983,
  "sourceName":"5GRAN_DU",
  "startEpochMicrosec":15198378,
  "timeZoneOffset":"UTC+05:00",
  "compression":"gzip",
  "fileFormatType":"org.3GPP.32.435#measCollec",
  "fileFormatVersion":"V10",
  "name":"5GRAN_DU/A20220418.1900-1915_seliitdus00487.xml",
  "changeIdentifier":"PM_MEAS_FILES",
  "objectStoreBucket":"ropfiles"
}
```


fig. 50. Output File Publish Message

Configuration

The DFC is configured via its application. Yaml

4.1.2.1 Non-RT RIC PM File Converter

Overview

The task of the PM File Converter is to convert PM Measurement report files, (XML formatted according to 3GPP TS 32.432 and 3GPP TS 32.435) into Json.

The component receives objects from kafka that indicates that new PM Report files are collected from the RAN. The event is sent by the Data File Collector.

The XML file is read from the storage, converted to Json, gzipped and stored.

A Json object indicating that a new Json PM Measurement report is available is sent on a Kafka topic to be picked up by other components for further processing

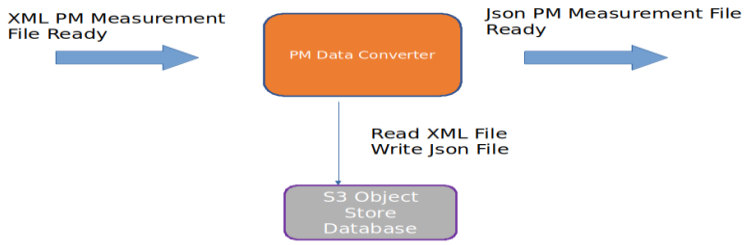


fig. 4-4.PM Data Converter workflow

Output PM measurement

The Json format of the PM measurement follows the same structure as the input XML format (defined by 3GPP).

Here follows an example:


```
{
   "event":{
      "commonEventHeader":{
         "domain":"perf3gpp",
         "eventId":"9efa1210-f285-455f-9c6a-3a659b1f1882",
         "eventName":"perf3gpp_gnb-Ericsson_pmMeasResult",
         "sourceName":"O-DU-1122",
         "reportingEntityName":"",
         "startEpochMicrosec":951912000000,
         "lastEpochMicrosec":951912900000,
         "timeZoneOffset":"+00:00"
      },
      "perf3gppFields":{
         "perf3gppFieldsVersion":"1.0",
         "measDataCollection":{
            "granularityPeriod":900,
            "measuredEntityUserName":"RNC Telecomville",
            "measuredEntityDn":"SubNetwork=CountryNN,MeContext=MEC-Gbg-1,ManagedElement=RNC-Gbg
            "measuredEntitySoftwareVersion":"",
            "measInfoList":[
               {
                  "measInfoId":{
                     "sMeasInfoId":"PM=1,PmGroup=NRCellDU_GNBDU"
                  },
                  "measTypes":{
                     "sMeasTypesList":[
                        "succImmediateAssignProcs"
                     ]
                  },
                  "measValuesList":[
                     {
                        "measObjInstId":"RncFunction=RF-1,UtranCell=Gbg-997",
                        "suspectFlag":"false",
                        "measResults":[
                           {
                              "p":1,
                              "sValue":"1113"
                           }
                        ]
                     },
                     {
                        "measObjInstId":"RncFunction=RF-1,UtranCell=Gbg-998",
                        "suspectFlag":"false",
                        "measResults":[
                           {
                              "p":1,
                              "sValue":"234"
                           }
                        ]
                     },
                     {
                        "measObjInstId":"RncFunction=RF-1,UtranCell=Gbg-999",
                        "suspectFlag":"true",
                        "measResults":[
                           {
                              "p":1,
                              "sValue":"789"
                           }
                        ]
                     }
                  ]
               }
            ]
         }
      }
   }
}
```


fig. 4-5.output pm measurement

Output file ready message

Here follows an example of the sent object indicating that a new Json file is available. It only contains the name of the stored file. The name of the bucket and the minio endpoint must be known by the event receiver.

```
{
   "filename": "xyz.json.gzip"
}
```

fig. 4-6.output file ready message

Input file ready message

Below follows an example of an input File Ready Message. The message is sent by the Data File

```
{
  "productName":"RnNode",
  "vendorName":"Ericsson",
  "lastEpochMicrosec":151983,
  "sourceName":"5GRAN_DU",
  "startEpochMicrosec":15198378,
  "timeZoneOffset":"UTC+05:00",
  "compression":"gzip",
  "fileFormatType":"org.3GPP.32.435#measCollec",
  "fileFormatVersion":"V10",
  "name":"5GRAN_DU/A20220418.1900-1915_seliitdus00487.xml",
  "changeIdentifier":"PM_MEAS_FILES",
  "objectStoreBucket":"ropfiles"
}
```

fig. 4-7.input file ready message

Collector. The only elements used by this component are sourceName, name and objectStoreBucket

4.1.2.2 Non-RT RIC PM Producer

Introduction

The task of the PM Producer is to process PM reports and to distribute requested information to subscribers. The main use case is:

- The PM Producer receives a Json object from Kafka which notifies that a new PM report is fetched and is available to processed.
- The actual PM report is in a file, which is stored in an S3 Object store bucket or in the file system (in a mounted volume). The file has the same structure as 3GPP TS 32.432/3GPP TS 32.435, but is converted to Json and is extended to contain the information that is encoded the 3GPP measurement report xml file name.
- The PM Producer loads the file and distributes the contents to the subscribers over Kafka according to their subscription parameters. These subscription parameters define wanted measurement types from given parts of the network.

The PM Producer registers itself as an information producer of PM measurement data in Information Coordination Service (ICS).

A data consumer can create an information job (data subscription) using the ICS consumer API (for Rapps) or the A1-EI (Enrichment Information) API (for NearRT-RICs). The PM Producer will get notified when information jobs of type 'PM measurements' are created.

The service is implemented in Java Spring Boot

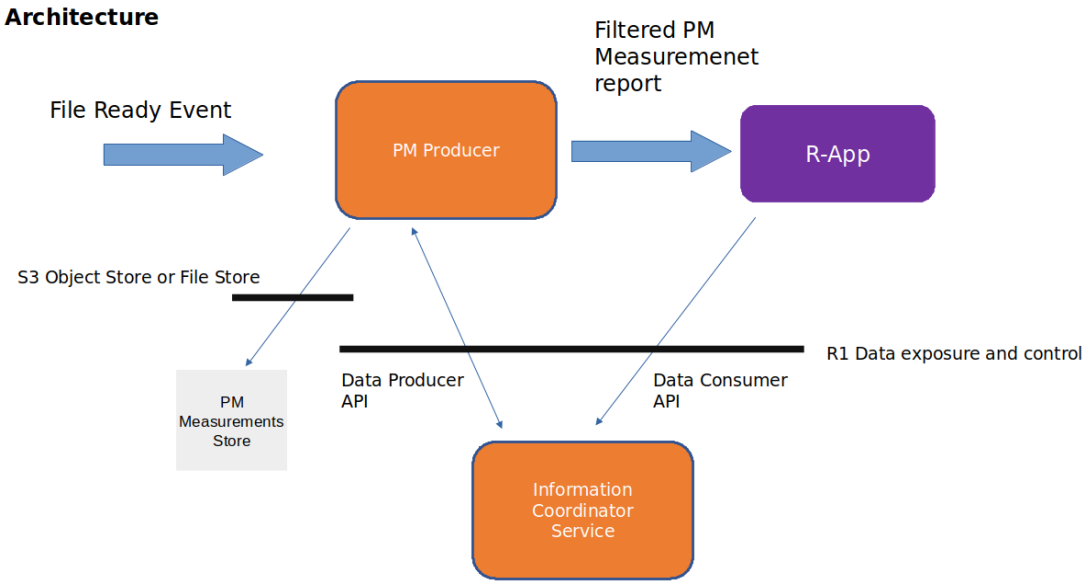


fig. 55. System Architecture for Performance Measurement Data Processing

Delivered data

When a data consumer (e.g. a Rapp) creates an Information Job, a Kafka Topic is given as output for the job. After filtering, the data will be delivered to the output topic.

The format of the delivered PM measurement is the same as the input format (which in turn is a Json mapping done from 3GPP TS 32.432/3GPP TS 32.435). The data can be delivered in gzipped format or in cleartext (indicated by an element in the Kafka header).

The result of the PM filtering preserves the structure of a 3GPP PM report. Here follows an example of a resulting PM report delivered.


```
{
   "event":{
      "commonEventHeader":{
         "domain":"perf3gpp",
         "eventId":"9efa1210-f285-455f-9c6a-3a659b1f1882",
         "eventName":"perf3gpp_gnb-Ericsson_pmMeasResult",
         "sourceName":"O-DU-1122",
         "reportingEntityName":"",
         "startEpochMicrosec":951912000000,
         "lastEpochMicrosec":951912900000,
         "timeZoneOffset":"+00:00"
      },
      "perf3gppFields":{
         "perf3gppFieldsVersion":"1.0",
         "measDataCollection":{
            "granularityPeriod":900,
            "measuredEntityUserName":"RNC Telecomville",
            "measuredEntityDn":"SubNetwork=CountryNN,MeContext=MEC-Gbg-1,ManagedElement=RNC-Gbg
            "measuredEntitySoftwareVersion":"",
            "measInfoList":[
               {
                  "measInfoId":{
                     "sMeasInfoId":"PM=1,PmGroup=NRCellDU_GNBDU"
                  },
                  "measTypes":{
                     "sMeasTypesList":[
                        "succImmediateAssignProcs"
                     ]
                  },
                  "measValuesList":[
                     {
                        "measObjInstId":"RncFunction=RF-1,UtranCell=Gbg-997",
                        "suspectFlag":"false",
                        "measResults":[
                           {
                              "p":1,
                              "sValue":"1113"
                           }
                        ]
                     },
                     {
                        "measObjInstId":"RncFunction=RF-1,UtranCell=Gbg-998",
                        "suspectFlag":"false",
                        "measResults":[
                           {
                              "p":1,
                              "sValue":"234"
                           }
                        ]
                     },
                     {
                        "measObjInstId":"RncFunction=RF-1,UtranCell=Gbg-999",
                        "suspectFlag":"true",
                        "measResults":[
                           {
                              "p":1,
                              "sValue":"789"
                           }
                        ]
                     }
                  ]
               }
            ]
         }
      }
   }
}
```


fig. 4-8.delievered data

4.1.2.3 Sent Kafka headers

For each filtered result sent to a Kafka topic, there will be the following properties in the Kafka header:

- type-id, this property is used to indicate the ID of the information type. The value is a string.
- gzip, if this property exists the object is gzip'ed (otherwise not). The property has no value.
- source-name, the name of the source RAN traffic-handling element from which the measurements originate.

4.1.2.4 Configuration

The component is configured by a configuration file and by the normal spring boot configuration file (application. yaml).

Configuration file

The configuration file defines Kafka topics that should be listened to and registered as information types which can be subscribed to. There is an example configuration file in config/application_configuration.json

Each entry will be registered as a subscribe information type in ICS. The following attributes can be used in each entry:

- id, the information type identifier.
- kafkaInputTopic, a Kafka topic to listen to for new file events.
- inputJobType, the information type for new file events subscription.
- inputJobDefinition, the parameters for the new file events subscription.

The last two parameters are used to create the subscription for the input to this component (subscription of file ready events).

Below follows an example of a configuration file.

```
{
  "types": [
     {
        "id": "PmDataOverKafka",
        "kafkaInputTopic": "FileReadyEvent",
        "inputJobType": "xml-file-data-to-filestore",
        "inputJobDefinition": {
           "kafkaOutputTopic": "FileReadyEvent",
           "filestore-output-bucket": "pm-files-json",
           "filterType": "pmdata",
           "filter": {
              "inputCompression": "xml.gz",
              "outputCompression": "none"
           }
        }
     }
  ]
}
```

4.1.2.5 application. yaml

As any spring boot application, this is component configured via an application.yaml file.

Information Job Parameters

The schema for the parameters for PM measurements subscription is defined in file src/main/resources/typeSchemaPmData.json.

4.1.2.6 typeSchemaPmData.json

The type specific Json schema for the subscription of PM measurement:

```
{
"$schema": "http://json-schema.org/draft-04/schema#",
"type": "object",
"additionalProperties": false,
"properties": {
    "filter": {
        "type": "object",
        "additionalProperties": false,
        "properties": {
            "sourceNames": {
                "type": "array",
                "items": [
                    {
                        "type": "string"
                    }
                ]
            },
            "measObjInstIds": {
                "type": "array",
                "items": [
                    {
                        "type": "string"
                    }
                ]
            },
            "measTypeSpecs": {
                "type": "array",
                "items": [
                    {
                        "type": "object",
                        "properties": {
                            "measuredObjClass": {
                                "type": "string"
                            },
                            "measTypes": {
                                "type": "array",
                                "items": [
                                    {
                                        "type": "string"
                                    }
                                ]
                            }
                        },
                        "required": [
                            "measuredObjClass"
                        ]
                    }
                ]
            },
            "measuredEntityDns": {
                "type": "array",
                "items": [
                    {
                        "type": "string"
                    }
                ]
            },
            "pmRopStartTime": {
                "type": "string"
            },
            "pmRopEndTime": {
                "type": "string"
            }
        }
    },
    "deliveryInfo": {
        "type": "object",
        "additionalProperties": false,
        "properties": {
            "topic": {
                "type": "string"
            },
            "bootStrapServers": {
                "type": "string"
            }
        },
        "required": [
            "topic"
        ]
    }
},
"required": [
    "filter", "deliveryInfo"
]
}
```

The following properties are defined:

filter, the value of the filter expression. This selects which data to subscribe for. All fields are optional and excluding a field means that everything is selected.
sourceNames, section of the names of the reporting RAN traffic-handling nodes
measObjInstIds, selection of the measured resources. This is the Relative Distinguished Name (RDN) of the MO that has the counter. If a given value is contained in the filter

definition, it will match (partial matching). For instance a value like “NRCellCU” will match “ManagedElement=seliitdus00487,GNBCUCPFunction=1,NRCellCU=32”.
measTypeSpecs, selection of measurement types (counters). This consists of
measuredObjClass, the name of the class of the measured resources.
measTypes, the name of the measurement type (counter). The measurement type name is only unique in the scope of an MO class (measured resource).
measuredEntityDns, selection of DNs for the RAN traffic-handling elements.
pmRopStartTime, if this parameter is specified already collected PM measurements files will be scanned to retrieve historical data. This is the time from when the information shall be returned. In this case, the query is only done for files from the given “sourceNames”. If this parameter is excluded, only “new” reports will be delivered as they are collected from the RAN traffic-handling nodes. How old information that can be retrieved depends on the retention time for the storage (if minio it used, it is a S3 bucket). A best effort is done, so that the stored files that are in time range are scanned even if the specified time is too long back in time.
pmRopEndTime, for querying already collected PM measurements. Only relevant if pmRopStartTime. If this parameters is given, no reports will be sent as new files are collected.deliveryInfo, defines where the subscribed PM measurements shall be sent.topic, the name of the kafka topic

bootStrapServers, reference to the kafka bus to used. This is optional, if this is omitted the default configured kafka bus is used (which is configured in the application.yaml file

below follows examples of some filters

```
{
  "filter":{
    "sourceNames":[
       "O-DU-1122"
    ],
    "measObjInstIds":[
       "UtranCell=Gbg-997"
    ],
    "measTypeSpecs":[
       {
          "measuredObjClass":"UtranCell",
```

```
{
  "filterType":"pmdata",
  "filter": {
    "sourceNames":[
       "O-DU-1122", "O-DU-1123"
    ],
```

fig. 4-9.a filter that will match two counters from all cells in two RAN

```
          "measTypes":[
             "pmCounterNumber0", "pmCounterNumber1"
          ]
       }
    ],

  }
}
```

fig. 4-10.traffic-handling nodes.

4.1.2.7 PM measurements subscription

The sequence is that a “new file event” is received (from a Kafka topic). The file is read from local storage (file storage or S3 object store). For each Job, the specified PM filter is applied to the data and the result is sent to the Kafka topic specified by the Job (by the data consumer).

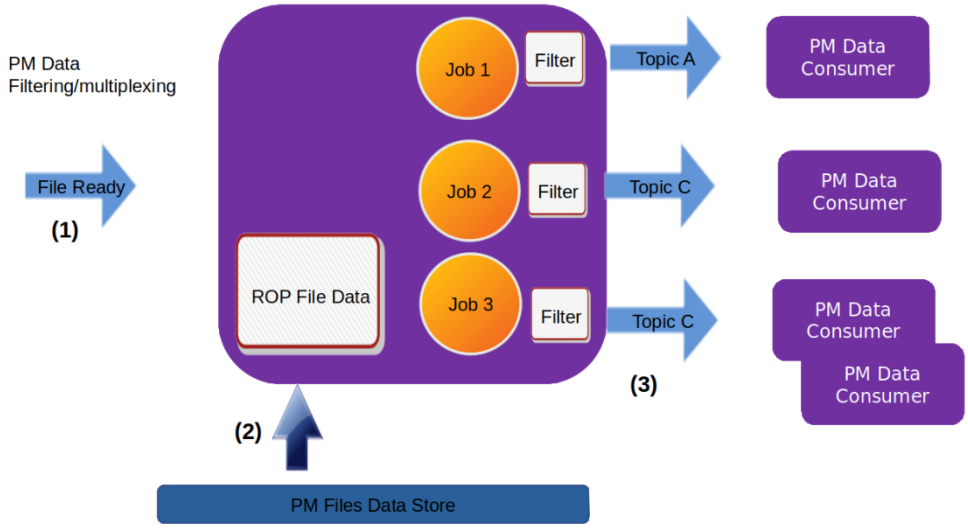


fig. 4-11.PM Data Filtering and Multiplexing Architecture

Several Jobs sharing the same Kafka topic

If several jobs are published to the same Kafka topic (shared topic), the resulting filtered output will be an aggregate of all matching filters. So, each consumer will then get more data than requested.

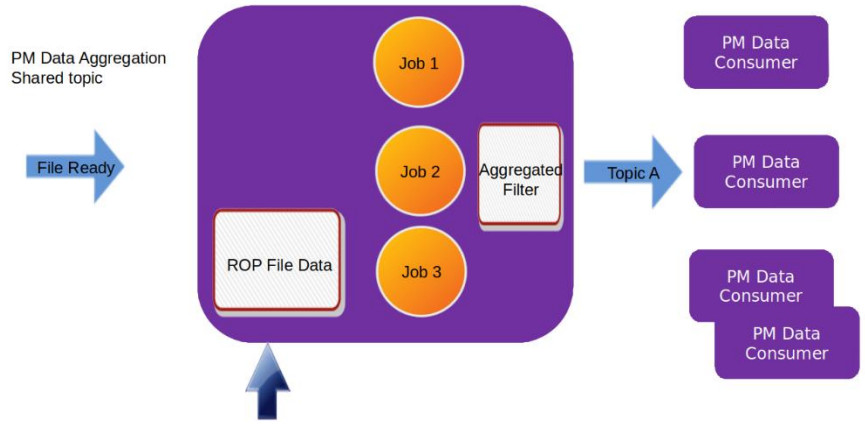


fig. 4-12. PM Data Aggregation Architecture

4.1.2.8 Non-RT RIC Influx Logger

Introduction

The task of the Influx Logger is to receive PM Measurement reports from a Kafka topic and to store the measurements in an Influx time series database.

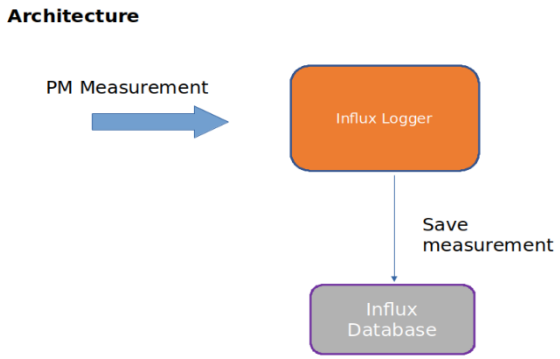


fig. 4-13. Influx Logging Architecture

The above product is a part of the NONRT-RIC

Database Schema

The PM Measurements are stored in the influx database. Each measured resource is mapped to a measurement in Influx. The name of the measurement is the Full Distinguished Name of the resource. Each measurement type (counter) is mapped to a field. The name of the field is the same as the name of the measurement type.

In addition, there is field which stores the GranularityPeriod in seconds of each report. The GP is for how long the counters have been aggregated.

The time is the end time of the report. The start time is then the logged time minus the granularity period.

Here follows an example of one Influx table which contains aggregated values for a measured resource.

Influx (Time Series Database)

Measurement:
SubNetwork=CountryNN,MeContext=MEC-Gbg-1,ManagedElement=RNC-Gbg-1,RncFunction=RF-1,UtranCell=Gbg-997

Time	attTCHSeizures5	succTCHSeizures	GranularityPeriod
2023-02-14T13:07:00.099Z	123	333	900
2023-02-14T13:07:51.637Z	456	444	900
2023-02-14T13:08:36.652Z	789	777	900

fig. 4-14.Influx table which contains aggregated values

Input PM Measurement

The PM measurement received from the Kafka topic is produced by the pm-producer. Here follows an example of the expected input object:


```
{
   "event":{
      "commonEventHeader":{
         "domain":"perf3gpp",
         "eventId":"9efa1210-f285-455f-9c6a-3a659b1f1882",
         "eventName":"perf3gpp_gnb-Ericsson_pmMeasResult",
         "sourceName":"O-DU-1122",
         "reportingEntityName":"",
         "startEpochMicrosec":951912000000,
         "lastEpochMicrosec":951912900000,
         "timeZoneOffset":"+00:00"
      },
      "perf3gppFields":{
         "perf3gppFieldsVersion":"1.0",
         "measDataCollection":{
            "granularityPeriod":900,
            "measuredEntityUserName":"RNC Telecomville",
            "measuredEntityDn":"SubNetwork=CountryNN,MeContext=MEC-Gbg-1,ManagedElement=RNC-Gbg
            "measuredEntitySoftwareVersion":"",
            "measInfoList":[
               {
                  "measInfoId":{
                     "sMeasInfoId":"PM=1,PmGroup=NRCellDU_GNBDU"
                  },
                  "measTypes":{
                     "sMeasTypesList":[
                        "succImmediateAssignProcs"
                     ]
                  },
                  "measValuesList":[
                     {
                        "measObjInstId":"RncFunction=RF-1,UtranCell=Gbg-997",
                        "suspectFlag":"false",
                        "measResults":[
                           {
                              "p":1,
                              "sValue":"1113"
                           }
                        ]
                     },
                     {
                        "measObjInstId":"RncFunction=RF-1,UtranCell=Gbg-998",
                        "suspectFlag":"false",
                        "measResults":[
                           {
                              "p":1,
                              "sValue":"234"
                           }
                        ]
                     },
                     {
                        "measObjInstId":"RncFunction=RF-1,UtranCell=Gbg-999",
                        "suspectFlag":"true",
                        "measResults":[
                           {
                              "p":1,
                              "sValue":"789"
                           }
                        ]
                     }
                  ]
               }
            ]
```


fig. 4-15.input pm measurement

Configuration

Setting up the PM measurement subscription

The influx logger will create its data subscription automatically. This is done by reading a configuration file that defines the data to log and which Kafka topic to use (1). The contents of this file is used to create the information job for subscribing of PM measurement (2). ICS will make sure that all PM Measurement producers are ordered to start producing data (3).

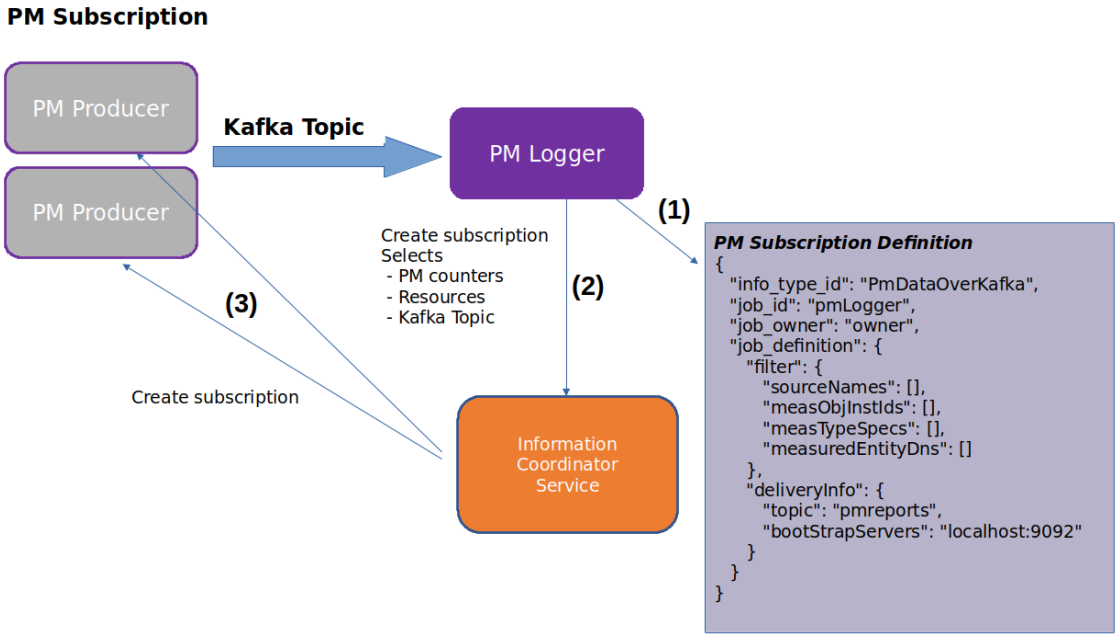


fig. 4-16.PM Subscription Architecture

application. yaml

The component is configured via its application. yaml

4.1.2.9 Information Coordination Service

The Information Coordination Service (ICS) is a generic service that maintains data subscriptions. Its main purpose is to decouple data consumers and data producers in a multi-vendor environment. A data consumer does not need to know anything about the producers of the data.

The following terms are used:

- Data Consumer is a subscriber of data. Subscription is done by creating an “Information Job”. A data consumer can for instance be an R-App (using the R1 API) or a NearRT-RIC consuming Enrichment Information (and uses the A1-EI API provided by this service).
- Information Type is a type of information. This defines an interface between consumers and producers. Each information type defines which parameters can be used for creating an information job. These parameters are defined by a Json schema connected to the Information Type. These parameters may **include:**

    - Parameters related to data delivery (for instance a callback URL if REST is used or a Kafka stream). These are different for different delivery protocols.
    - Filtering or other information for data selection.
    - Periodicy
    - Other info used for aggregation
- Data Producer is a producer of data. A producer will get notified about all information jobs of its supported types. This also means that filtering is done at the producer (ideally at the source of the data). A data producer can for instance be an R-App.

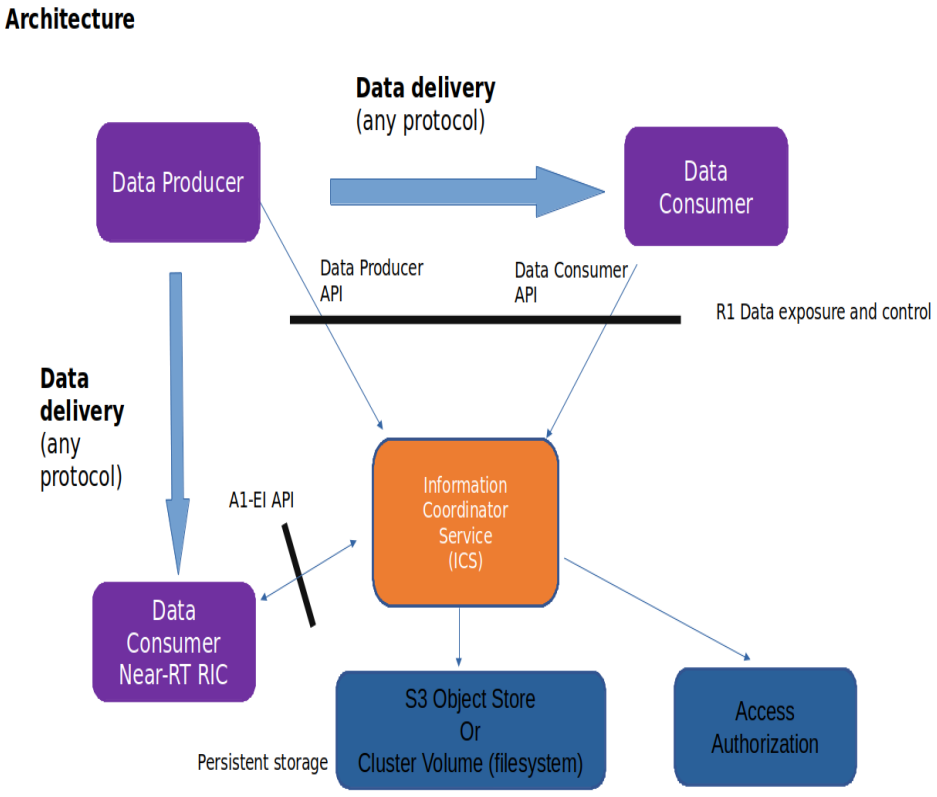


fig. 4-17.Data Coordination Architecture

One information type can be supported by zero to many data producers and can be subscribed to by zero to many data consumers. For example, there can be two data producers for a type of data: one

from one vendor (handling a part of the network) and another from a different vendor. A data consumer is agnostic about this.

Information Jobs and types are stored persistently by ICS in a local database. This can be either using Amazon S3 - Cloud Object Storage or file system.

To restrict which data that can be consumed and by whom there is support for fine-grained access control. When data subscriptions/jobs are modified or read, an access check can be performed. ICS can be configured to call an external authorizer. For example, this can be an Open Policy Agent (OPA) which can grant or deny accesses based on an access token (JWT) used by the calling data consumer. In addition to this the information type, access type (read/write) and all type specific parameters can be used by access rules.

The URL to the authorization component is defined in the application.yaml file and the call invoked to by ICS is described in API documentation.

Key Requirements

- Multiple producers, possibly from different vendors, should be able to produce the same type of data.
- The system should allow for arbitrary installation, upgrade, scaling up, scaling down, and restarting software, in any order.
- ICS should decouple data producers from data consumers. Consumers should not be aware of producers, their endpoints, or the number of producers. Upgrades, scaling, and other changes to producers should not affect consumers. That also means that a producer can be replaced by another one.
- Data consumers should be able to start subscriptions at any time, including before a producer is installed, or during a producer’s upgrade, restart, or scaling up/down.
- The system should impose as few restrictions as possible on how software is implemented. Applications may, for example, be implemented as multiple identical executing entities for load sharing.
- The lifecycle of a data type in the system should not be controlled by a single producer. There may be data types and subscriptions for data types that have no producers, such as during an upgrade scenario.
- A data specification should contain all the information needed to implement both producers and consumers. The data type specification forms the API between these. It must define the ID of the data type, all possible parameters used at subscription creation (such as filter constructions), and all details about the delivered format and the delivery mechanism. This is comparable to a Managed Object Model.
- ICS should mediate data type versioning and choose compatible data producers. For example, if there are two producers, one producing datatype version 1.5.0 and another producing version 1.4.0, and a consumer subscribes to data version 1.1.0, both these producers should be involved (since version 1.4.0 and version 1.5.0 are backwards compatible with version 1.1.0). The system should be able to handle these versioning scenarios automatically.

- A consumer shall be able to retrieve its jobs after a restart. Therefore it must be possible to group the jobs based on a label “owner” which is defined by the consumer. This must be unique, which suggests that it should be based on UUIDS, but this is up to the consumer. This “owner” can be a POD, an application etc. ICS should not restrict that.

Summary of principles

- ICS provides APIs for control of data subscriptions but is not involved in the delivery of data. This means that any delivery protocol can be used.
- Data for one Information type can be produced by many producers from different vendors.
- Data filtering is done by the producer. ICS does not restrict how data selection/filtering is done.
- A Data Consumer can create a data subscription (Information Job) regardless of the status of the data producers. The producers can come and go without any need for the Data Consumer to take any action. A subscription indicates the need for a type of data and the system should do its best to fulfill this.
- ICS is by design not aware of any subscribe data types.
- When a consumer creates a subscription/job, ICS shall choose the information type version with the lowest available compatible version. All producers that have registered a type that is compatible with the chosen version are included. Example, if a consumer creates a job with type version 1.1.0, the chosen type may 1.2.0 and a producer supporting version 1.9.0 will be included (but not a producer that supports version 2.0.0).

In the example, there is one subscription, and the type of data is supported by two producers. That means that both producers are aware of the information job and will delver data directly to the subscriber.

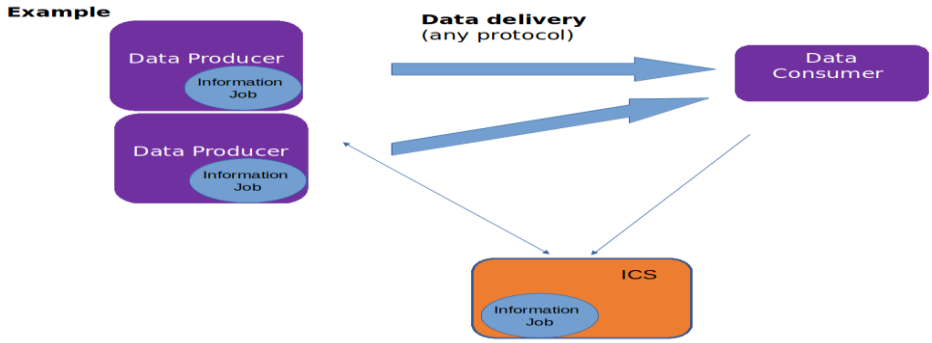


fig. 4-18.Data Delivery Example Architecture

So, a typical sequence is that:

- An Information Type is registered.
- Producers of the Information Types are registered
- A Consumer creates an Information Job of the type and supplies the type specific parameters for data delivery and filtering etc.
- The producers get notified of the job and will start producing data.

If a new producer is started, it will register itself and will get notified of all jobs of its supported types.

4.1.3 implementation

Implemented as a java spring boot application

Configuration:

The component is configured by the usual spring boot application.yaml file.

An example application. yaml configuration file

Data Flow:

The figure below gives an overview of the data flow through the components.

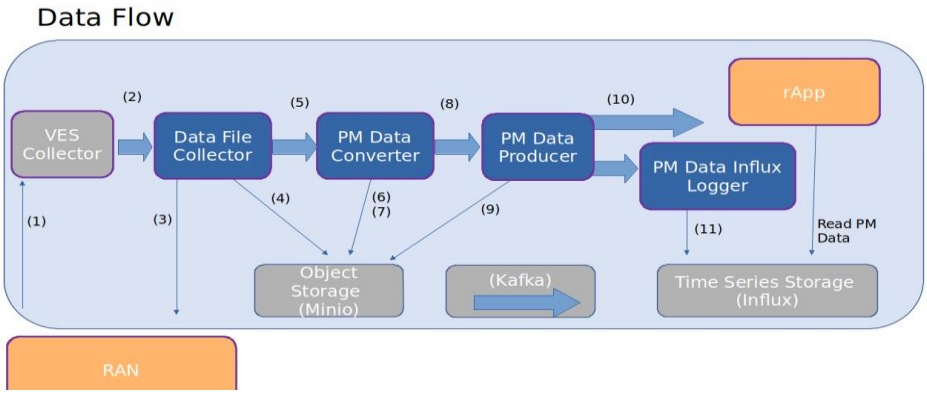


fig. 4-19. PM Data Flow Architecture

1. **The RAN node sends a VES event with available PM measurement report files.**
2. **The VES event is put on a Kafka topic and picked up by the Data File Collector.**
3. **A PM report file is fetched from the RAN node by a file transfer protocol. Which protocol to use is defined in the VES event.**
4. **The collected file is stored**
5. **A File collected object is put on a Kafka topic and is picked up by the PM File Converter.**

6. **The file data is read from the file store.**
7. **A PM report in Json format is stored (compressed with gzip).**
8. **A message (a Json object) indicating that a new PM report (in Json format) is available is put on a Kafka topic and is picked up by the PM Data Producer.**
9. **The PM data producer reads the Json file**
10. **The subscribed PM data is sent to the PM data consumers (over Kafka). An Rapp may be a PM data consumer.**
11. **The Influx Logger, which is a PM data consumer, stores PM data in an Influx database.**

At any time an Rapp can read logged PM data from the Influx database.

PM Data subscription:

PM measurement data is subscribed by creating an Information Job using the Information Coordination Service (ICS). This a subscription broker and is part of what is called Data Management an Exposure (DME) in O-RAN. The ICS makes sure that all data producers get its data subscriptions (jobs).

In the picture below, a Rapp and the Influx Logger are consumers of PM data.

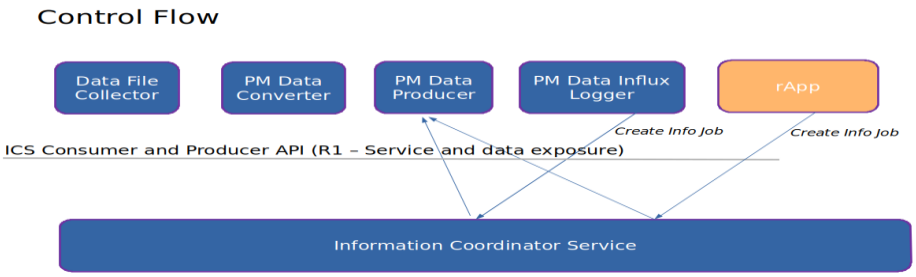


fig. 4-20.PM Control Flow Architecture

The PM Data Influx logger will create a PM data subscription based on a configuration file. A Rapp can create PM data subscription. The PM Data producer will deliver received PM measurements according to the subscriptions.

The PM Data file collector will fetch all PM measurement files. The PM Data Converted will convert all fetched xml files to Json. So these do not use any subscriptions.

PM Subscriber design time dependencies

An Rapp uses the ICS API to create and manage the subscription of PM Measurements.

The schema for the PM Measurement information jobs is defined in Non-RT RIC - RAN PM - PM Producer (Documentation. This schema defines parameters used in the subscription (info job) and defines which measurements to subscribe for and on which Kafka topic the information shall be delivered to.

An application retrieving logged PM data from the Influx database needs to consider how the data is stored (the schema). That is defined in Non-RT RIC - RAN PM - Influx Logger (Documentation).

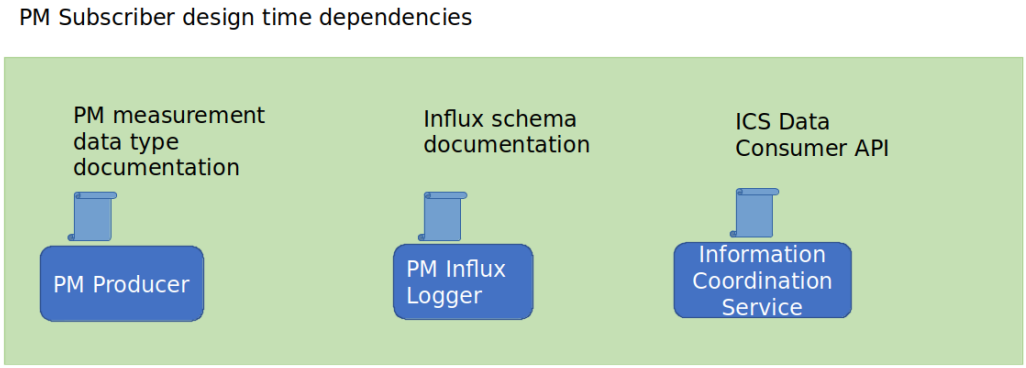


fig. 4-21.PM Subscriber Design Time Dependencies

4.1.3.1 Apache Kafka – A Distributed Event Streaming Platform

Introduction

Apache Kafka is a powerful open-source distributed system designed for real-time event streaming at scale. It serves as a robust messaging backbone for modern data architectures, enabling the continuous ingestion, storage, and processing of high-throughput data streams. Originally developed by LinkedIn and now maintained by the Apache Software Foundation,

Kafka is engineered for fault tolerance, scalability, and low-latency data delivery. It plays a critical role in enabling event-driven systems, stream processing, and decoupled microservice communication.

4.1.3.2 Fundamental Concepts

Kafka's architecture revolves around five primary elements:

Topics: Named channels where messages are published. Topics serve as logical containers for streaming data.

Partitions: Each topic is divided into one or more partitions. A partition is an ordered, immutable sequence of records and forms the unit of parallelism in Kafka.

Producers: Applications or services that publish data to Kafka topics.

Consumers: Applications that subscribe to topics and process the incoming data.

Brokers: Kafka servers responsible for storing and managing partitions. A Kafka cluster typically consists of multiple brokers to ensure load distribution and resilience.

Kafka ensures message ordering within each partition and maintains an offset for each record, enabling consumers to resume processing from any point in the stream. Consumers can also be grouped into consumer groups to enable parallel and distributed data processing across multiple instances

System architecture

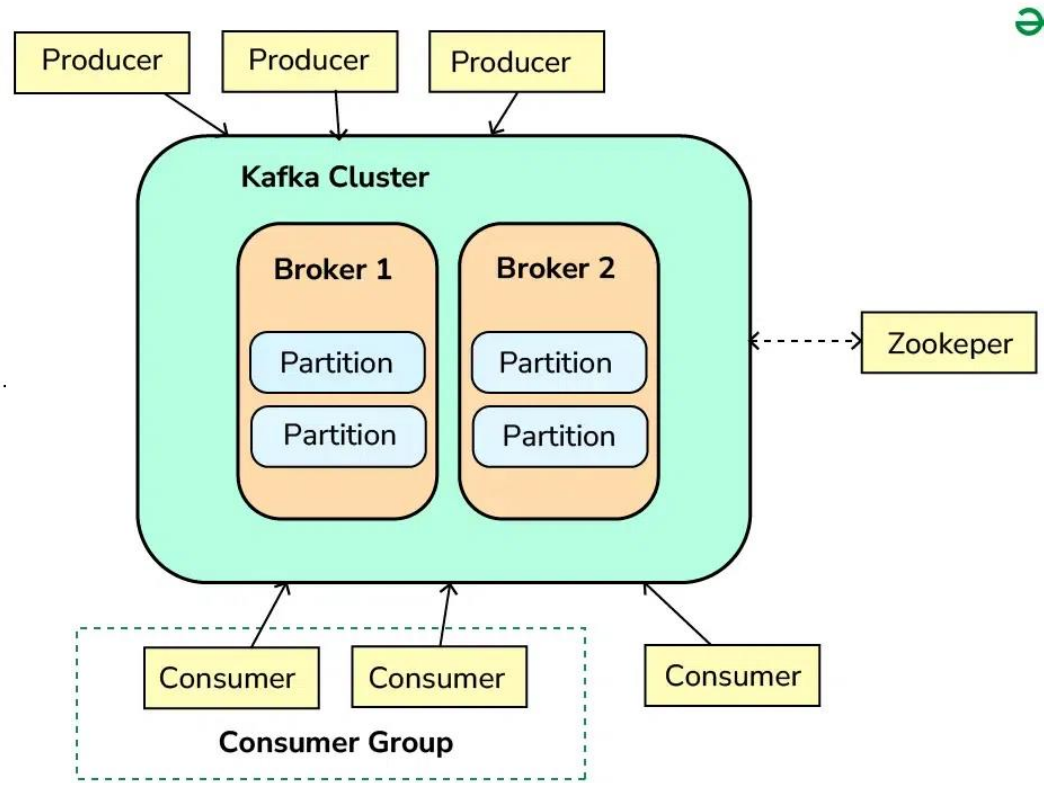


fig. 4-22.kafka system architecture

Kafka operates as a distributed cluster of broker nodes. Each partition of a topic is assigned to one broker as the *leader*, while one or more *replica* brokers serve as backups. This architecture ensures high availability and fault tolerance: if the leader broker becomes unavailable, one of the replicas is promoted to lead automatically.

Data produced to Kafka is written to disk and replicated across brokers. Consumers connect to the cluster and pull data directly from the partition leaders. Kafka uses a highly efficient binary protocol, and its design—based on sequential disk writes and zero-copy transfer—achieves exceptionally low latency and high throughput.

Kafka's ecosystem includes:

- **Kafka Connect: A tool for integrating Kafka with external systems (e.g., databases, cloud services).**

- **Kafka Streams: A lightweight stream-processing library built on top of Kafka's core APIs.**

Kafka can scale horizontally by adding more brokers to the cluster, allowing it to handle millions of messages per second.

4.1.3.3 Common Use Cases

Apache Kafka is widely adopted across industries due to its flexibility and robustness. Common scenarios include:

- **Real-Time Data Streaming: Kafka is ideal for processing continuous data flows such as clickstreams, financial transactions, and IoT sensor data.**

- **Log Aggregation: Kafka enables centralized log collection from various applications and servers, facilitating real-time monitoring and alerting.**

- **Event Sourcing: Kafka acts as a durable event store, recording every change as a time-stamped event. This allows systems to reconstruct application state from event history.**

- **Decoupled Microservices: Kafka facilitates asynchronous communication between services, improving modularity and reducing inter-service dependencies.**

Advantages of Apache Kafka

Apache Kafka offers several compelling benefits:

- **Scalability: Kafka clusters scale easily by distributing partitions across multiple brokers.**

- **Durability: Data is written to disk and replicated across nodes to ensure persistence.**

- **Fault Tolerance: Kafka automatically handles broker failures and ensures data availability.**

- **High Throughput: Kafka is optimized to process millions of messages per second with minimal latency.**

- **Message Replay: Consumers can reprocess data by resetting their offset, enabling error recovery and audit trails.**

- **Loose Coupling: Kafka decouples data producers from consumers, enhancing flexibility and modular system design.**

Conclusion

Apache Kafka has become a cornerstone of real-time data infrastructure in modern software systems. Its combination of durability, performance, and architectural flexibility makes it an ideal solution for streaming analytics, large-scale data pipelines, and event-driven architectures. As data continues to grow in volume and velocity, Kafka stands out as a resilient, scalable, and production-ready platform for processing real-time information efficiently.

4.1.3.4 Deploying Ranpm Using Kubernetes

The deployment of the Radio Access Network Performance Management (RANPM) system is a critical step in enabling intelligent and energy-efficient network management within the O-RAN (Open Radio Access Network) ecosystem. Building on the energy-saving strategies explored in the previous chapter, such as RF Channel Reconfiguration and Cell On/Off Switching, this section details the process of deploying RANPM within a Kubernetes

environment using Rancher Kubernetes Engine (RKE) version 1.8.3. The deployment leverages Rancher's robust management capabilities to orchestrate a scalable and resilient Kubernetes cluster, hosting key RANPM components like pm-https-server, pm-Rapp, Apache Kafka, and Influx DB. This setup ensures the collection, processing, and streaming of performance management (PM) data to support AI-driven optimization through integration with the Artificial Intelligence and Machine Learning Framework (AIMLFW). This section begins by introducing Rancher, followed by a step-by-step

guide to deploying RANPM, verifying the setup, and addressing potential challenges

4.1.3.5 Introducing Rancher

what is rancher?

Rancher is an open-source Kubernetes management platform developed by SUSE, designed to simplify the deployment, management, and operation of Kubernetes clusters across diverse environments. It provides a centralized interface for managing multiple Kubernetes clusters, offering features such as role-based access control (RBAC), monitoring, logging, and integration with continuous integration/continuous deployment (CI/CD) pipelines. Rancher supports deploying Kubernetes on bare metal, virtual machines (VMs), or cloud providers, making it a versatile choice for production-grade deployments.

Rancher uses the Rancher Kubernetes Engine (RKE), a lightweight and certified Kubernetes distribution, to provision and manage clusters. RKE v1.8.3, used in this project, offers a streamlined installer that configures Kubernetes with production-ready features, including high availability, security, and scalability. Additionally, Rancher supports RKE2, a hardened Kubernetes distribution for enhanced security, but RKE v1.8.3 was selected for its compatibility with the RANPM deployment requirements specified in the O-RAN Software Community (O-RAN SC) documentation.

4.1.3.6Why use rancher for RANPM?

Rancher was chosen to deploy the Kubernetes cluster in this project due to its key advantages:

- **Multi-Cluster Management: Rancher's centralized dashboard allows efficient management of multiple Kubernetes clusters, which is beneficial for scaling RANPM deployments across different environments.**

- **Ease of Use: RKE simplifies cluster provisioning with a single configuration file (cluster.yml), reducing setup complexity compared to other Kubernetes distributions.**
- **Production-Grade Features: RKE v1.8.3 supports high availability, RBAC, and integration with tools like Helm, ensuring a robust environment for RANPM components.**
- **Flexibility: Rancher's ability to deploy Kubernetes on various infrastructures (bare metal, VMs, or cloud) aligns with the O-RAN goal of open and interoperable systems.**
- **Monitoring and Logging: Rancher provides built-in tools for monitoring cluster health and logging, critical for troubleshooting RANPM deployments and ensuring reliable performance data collection.**

By leveraging Rancher and RKE v1.8.3, the RANPM system is deployed in a scalable and resilient Kubernetes environment, capable of handling the high-throughput data demands of 5G networks and supporting energy-efficient strategies through real-time performance analytics.

Setting up the Kubernetes Cluster with Rancher

The first step in deploying RANPM is establishing a Kubernetes cluster using RKE v1.8.3. The following outlines the initial setup process:

4.1.3.7Rancher prerequisites

Before deploying the cluster, ensure the following dependencies are installed:

Docker: Used to run containerized RANPM components. Install Docker on all nodes that will form the Kubernetes cluster.

RKE CLI: The RKE command-line tool (version 1.8.3) is required to provision the cluster. Download and install it from the Rancher releases page.

Node Requirements: At least one node for the control plane, with sufficient CPU, memory, and storage to handle RANPM workloads. Additional nodes can be added for worker roles.

Networking: Configure network access for ports used by RANPM services (e.g., 31784, 31823, 31767) to enable external communication

4.1.3.7RKE Cluster Configuration

RKE uses a configuration file (cluster.yml) to define the cluster's topology and settings. A sample configuration for RANPM includes:

- **Nodes: Specify the control-plane node and optional worker nodes, with roles defined (e.g., controlplane, worker).**
- **Port Mappings: Configure ports like 31784 (for pm-https-server), 31823 (for Non-RT RIC DME), and 31767 to ensure accessibility.**
- **Services: Enable Kubernetes services like kube-apiserver, kube-controller, and kubelet, tailored for RANPM requirements.**

Example cluster.yml snippet:

```
nodes:
 - address: <node-ip>
   user: <user>
   role: [controlplane, worker]
   ssh_key_path: ~/.ssh/id_rsa
services:
 kube-api:
   extra_args:
    feature-gates: "SomeFeature=true"
network:
 plugin: canal
```

Run the following command to deploy the cluster:

```
rke up --config cluster.yml
```

This command provisions the Kubernetes cluster, generating a kubeconfig file for accessing the cluster.

4.1.3.8 Accessing the cluster

After deployment, copy the kubeconfig file to the user's home directory to enable cluster access:

```
sudo cp /root/.kube/config $HOME/.kube/config
sudo chown $USER:$USER $HOME/.kube/config
```

This allows interaction with the cluster using kubectl commands, facilitating the deployment of RANPM components.

4.1.3.9 Next steps in the RANPM Deployment

With the Kubernetes cluster established, the deployment proceeds with:

- **Installing Istio: Deploy Istio v1.23.2 to manage service communication, using the istioctl tool.**
- **Building and Loading RANPM Images: Build Docker images for pm-https-server and pm-Rapp from the RANPM repository and load them into the RKE cluster.**
- **Deploying RANPM Components: Use Helm charts to deploy components like Apache Kafka, InfluxDB, and pm-Rapp, as detailed in the O-RAN SC documentation.**
- **Verification: Check pod status in the nonrtric and ran namespaces to ensure all components (e.g., kafka-1-kafka-0, influxdb2-0) are running correctly.**

These steps ensure a fully functional RANPM system, capable of collecting and processing performance data for energy optimization

4.1.3.10 Significance of Deployment

The deployment of RANPM using Rancher and RKE v1.8.3 establishes a robust and scalable environment for performance management. By integrating with AIMLFW, the system enables AI-driven analytics to optimize energy consumption, supporting use cases like RF Channel Reconfiguration by streaming real-time performance metrics (e.g., pdcpBytesDl, pdcpBytesUl) through Kafka to AI/ML models. This deployment aligns with the O-RAN vision of open, intelligent, and energy-efficient networks, providing a practical foundation for the next-generation wireless communication systems explored in this project.

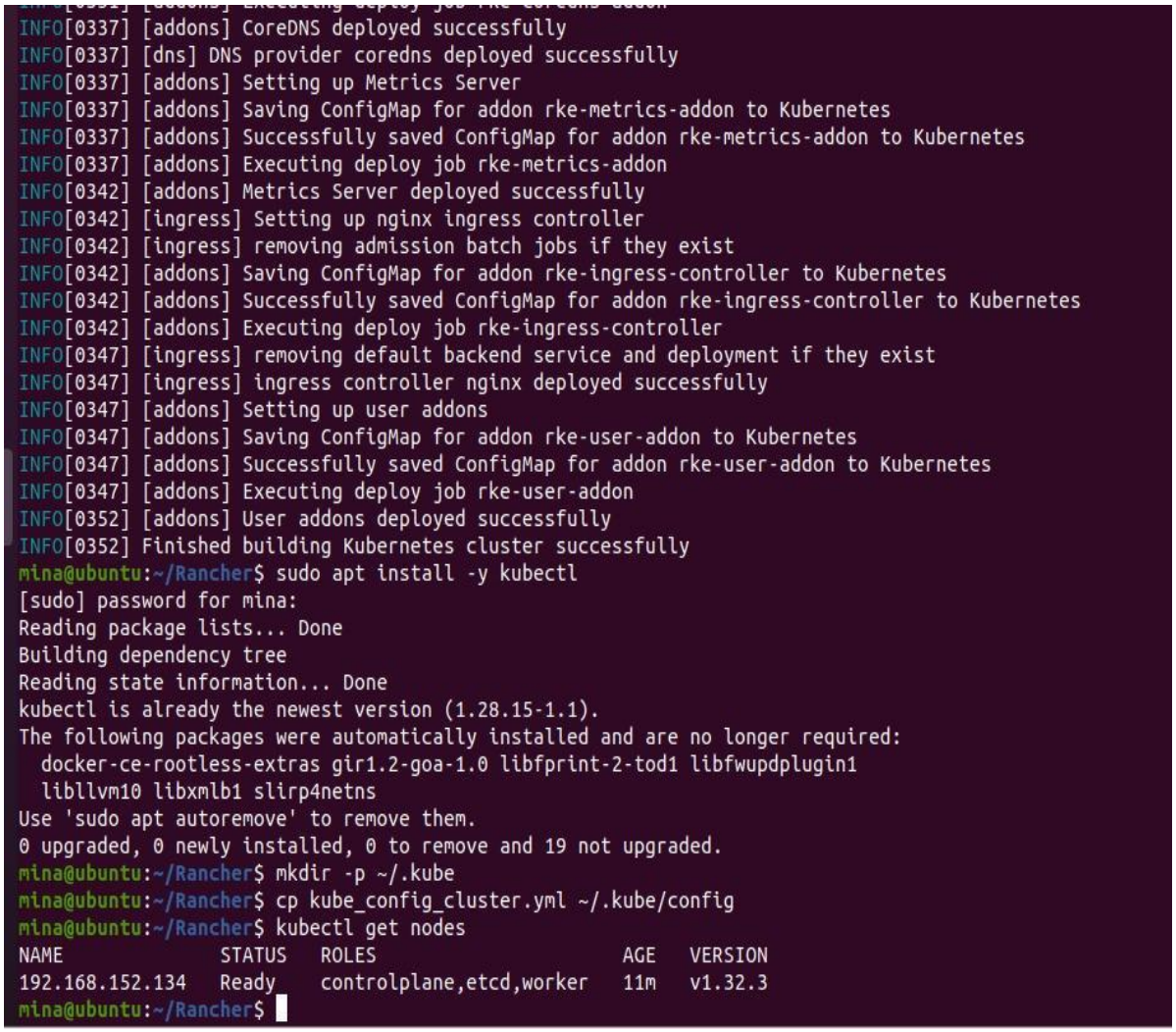


fig. 4-23. Kubernetes Deployment Log

4.1.3.11Prerequisites for RANPM Deployment

To deploy the Radio Access Network Performance Management (RANPM) system on a Linux, macOS, or Windows (via WSL) environment using a local or remote Kubernetes cluster, the following prerequisites are required to ensure a robust setup:

- **Kubernetes Cluster: A Kubernetes cluster provisioned with Rancher Kubernetes Engine (RKE) v1.8.3, with kubectl configured to access the cluster. Nodeports (e.g.,**

31784, 31823, 31767) must be accessible from the local machine to enable communication with RANPM services.

- **Docker: Installed locally to build RANPM images (e.g., ran-https-server:latest, pm-Rapp:latest) using the provided build scripts.**
- **Istio: Version 1.23.2 installed to manage service communication within the Kubernetes cluster.**
- **Additional Tools:**
  - **Helm 3: For deploying RANPM components via Helm charts, enabling automated and reproducible deployments.**
  - **Bash: For executing installation and build scripts, such as build.sh and install-nrt.sh.**
  - **envsubst: For environment variable substitution in configuration files (verify with type envsubst).**
  - **jq: For JSON processing, essential for handling PM data (verify with type jq).**
  - **keytool: For managing certificates used in secure communication.**
  - **openssl: For configuring secure communication protocols.**
- **Image Repository: For remote or multi-node clusters, an external container registry is required, configured in helm/global-values.yaml with a trailing / (e.g., extimagerepo: myrepo/). For local deployments, images are used directly without pushing to a registry.**

Significance: These prerequisites establish a robust environment for RANPM deployment, supporting the high resource demands of 5G performance management and enabling integration with the Artificial Intelligence and Machine Learning Framework (AIMLFW) for energy optimization strategies, such as RF Channel Reconfiguration and Cell On/Off Switching.

4.1.3.12 Setting up the Kubernetes cluster with Rancher

The deployment begins with provisioning a Kubernetes cluster using RKE v1.8.3. A cluster.yml file defines node roles (e.g., controlplane, worker) and port mappings (e.g., 31784, 31823, 31767) to ensure accessibility of RANPM services. After provisioning, the kubeconfig file is copied to the user's home directory to enable cluster access via kubectl:

```
sudo cp /root/.kube/config $HOME/.kube/config

sudo chown $USER:$USER $HOME/.kube/config
```

Significance: This step establishes the Kubernetes environment, providing a scalable and resilient platform for deploying RANPM components and supporting real-time performance data processing.

4.1.3.13 Installing RANPM Components

The RANPM deployment is orchestrated using scripts and Helm charts, as outlined in the O-RAN Software Community (O-RAN SC) documentation. The process involves building custom container images for pm-https-server and pm-Rapp, deploying core components, and setting up additional functionalities to support performance management.

4.1.3.14 Building RANPM Images

Custom container images for pm-https-server and pm-Rapp are built locally using the provided build.sh script with the no-push option, ensuring images are available to the RKE cluster without requiring a push to an external registry.

4.1.3.15Building pm-https-server image

The pm-https-server securely receives PM reports via HTTPS, serving as the entry point for performance data in the RANPM system.

Steps:

Clone the RANPM Repository: Clone the source code from the O-RAN SC repository to access the pm-https-server code.
git clone "https://gerrit.o-ran-sc.org/r/nonrtric/plt/ranpm"

1. **Navigate to the pm-https-server Directory: Access the directory containing the pm-https-server source code and build script.**
   cd ~/nonrtric-plt-ranpm/https-server
2. **Build the Container Image: Execute the build script with the no-push option to create the ran-https-server:latest image locally.**
   ./build.sh no-push

3. **Verify the Image: Confirm the image is built and available in the local environment.**
   docker images ran-https-server:latest
4. **Alternatively, verify the image is used by the deployed pods in the RKE cluster:**
   kubectl get pods -n nonrtric -o

Significance: The pm-https-server enables secure ingestion of PM reports (e.g., XML files), forwarding them to pm-Rapp for processing. Multiple instances (e.g., pm-https-server-0 to pm-https-server-9) ensure scalability and load balancing. The no-push option streamlines the build process for local Kubernetes deployments, with Helm charts managing image integration into the RKE cluster.

4.1.3.16 Building pm-Rapp Image

The pm-Rapp processes PM data within the Non-RT RIC ecosystem, converting data formats and publishing to Kafka topics.

Steps:

Navigate to the pm-Rapp Directory: Access the directory containing the pm-Rapp source code and build script.
cd ~/nonrtric-plt-ranpm/pm-Rapp

1. **Build the Container Image: Execute the build script with the no-push option to create the pm-Rapp:latest image locally.**
   ./build.sh no-push

2. **Verify the Image: Confirm the image is built and available.**
   docker images pm-Rapp:latest
3. **Significance: The pm-Rapp converts PM data (e.g., XML to JSON) and publishes it to Kafka topics like pmreports, enabling integration with AIMLFW for AI-driven energy optimization, such as RF Channel Reconfiguration and Cell On/Off Switching.**

4.1.3.17 Deploying RANPM core components

The core RANPM components, including pm-Rapp, Apache Kafka, and InfluxDB, are deployed using a main installation script that leverages Helm charts.

Steps:

Navigate to the Installer Directory: Access the directory containing Helm charts and installation scripts.
cd ~/nonrtric-plt-ranpm/install

1. **Configure Helm Values: Edit helm/global-values.yaml to specify parameters, such as the external image repository (if used) or local image settings.**
   nano helm/global-values.yaml
2. **Run the Main Installation Script: Execute the install-nrt.sh script to deploy core RANPM components in the nonrtric and ran namespaces.**
   ./install-nrt.sh

Significance: The install-nrt.sh script automates the deployment of RANPM components, establishing the infrastructure for processing and streaming PM data to support energy-saving strategies in 5G networks.

Components overview:

- **Redpanda Console:**

Nodeport: 31767

Purpose: Monitors Kafka topics and consumers, replacing traditional Kafka consoles for real-time data flow visualization in RANPM.

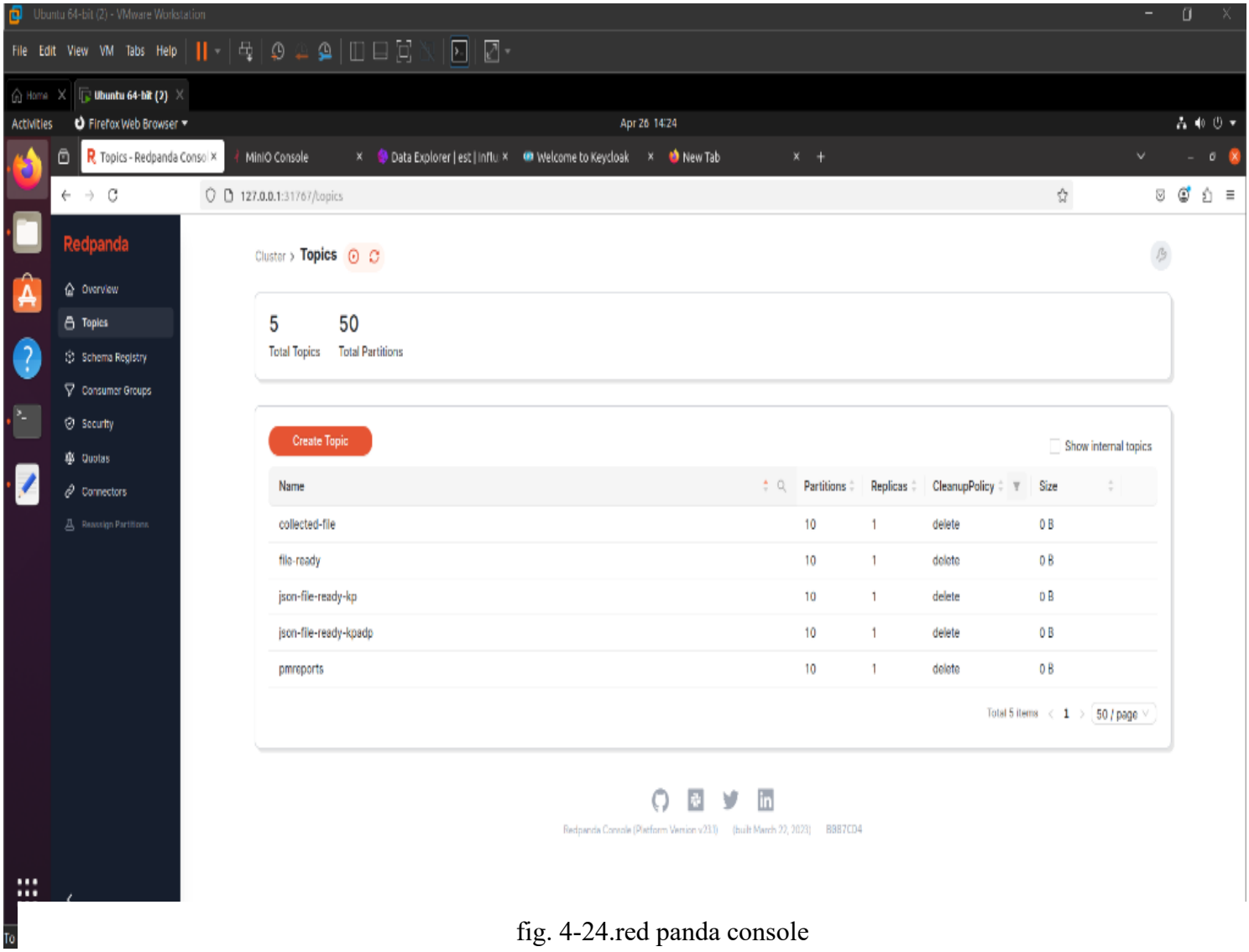


fig. 4-24.red panda console

- **Keykloack:**

Keycloak Admin Console:

Nodeport: 31788

Credentials: User: admin, Password: admin

Purpose:Manages authentication.

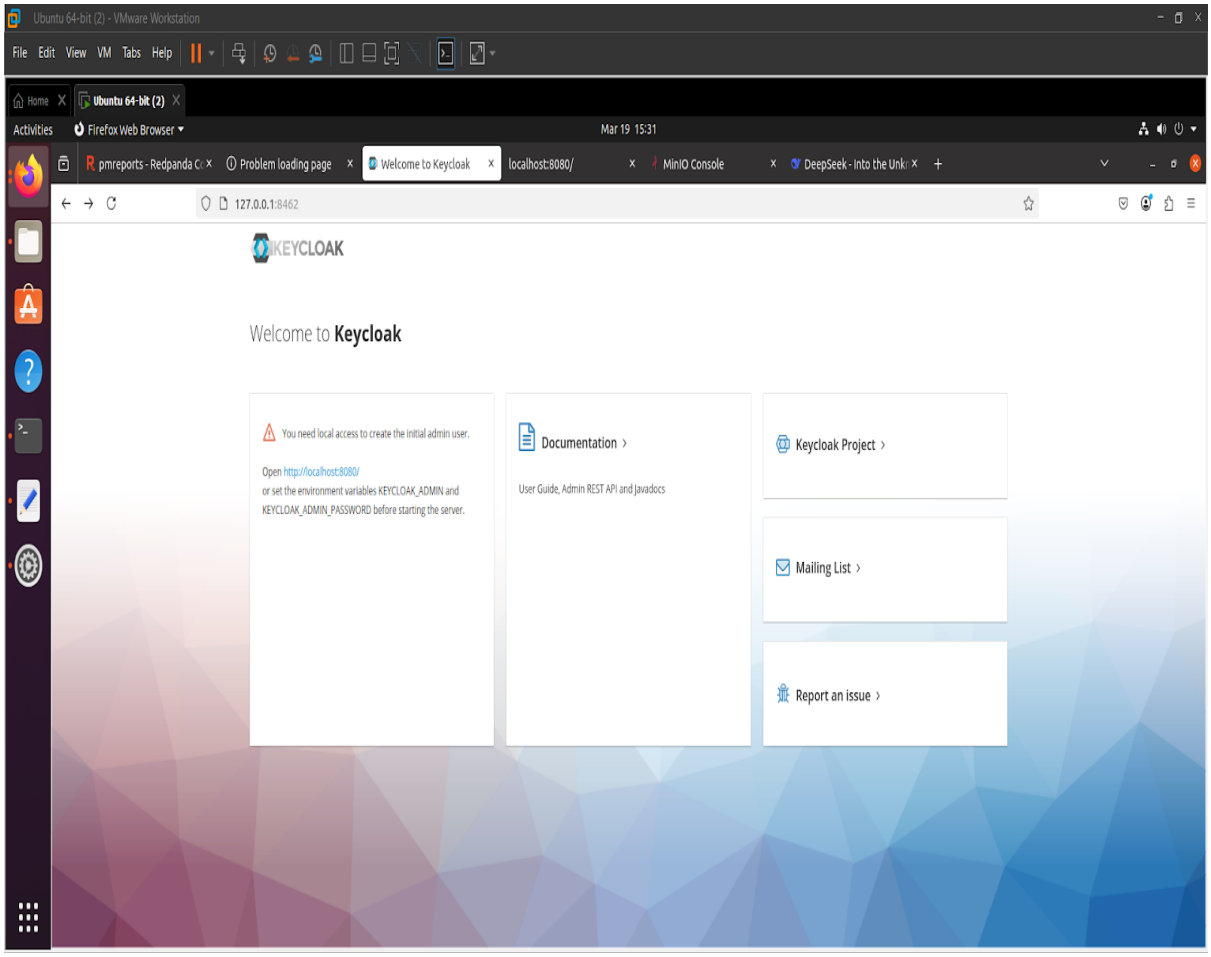


fig. 4-25.key cloak console

Minio Web Interface:

Nodeport: 31768

Credentials: User: admin, Password: adminadmin

Purpose: Manages PM report storage.

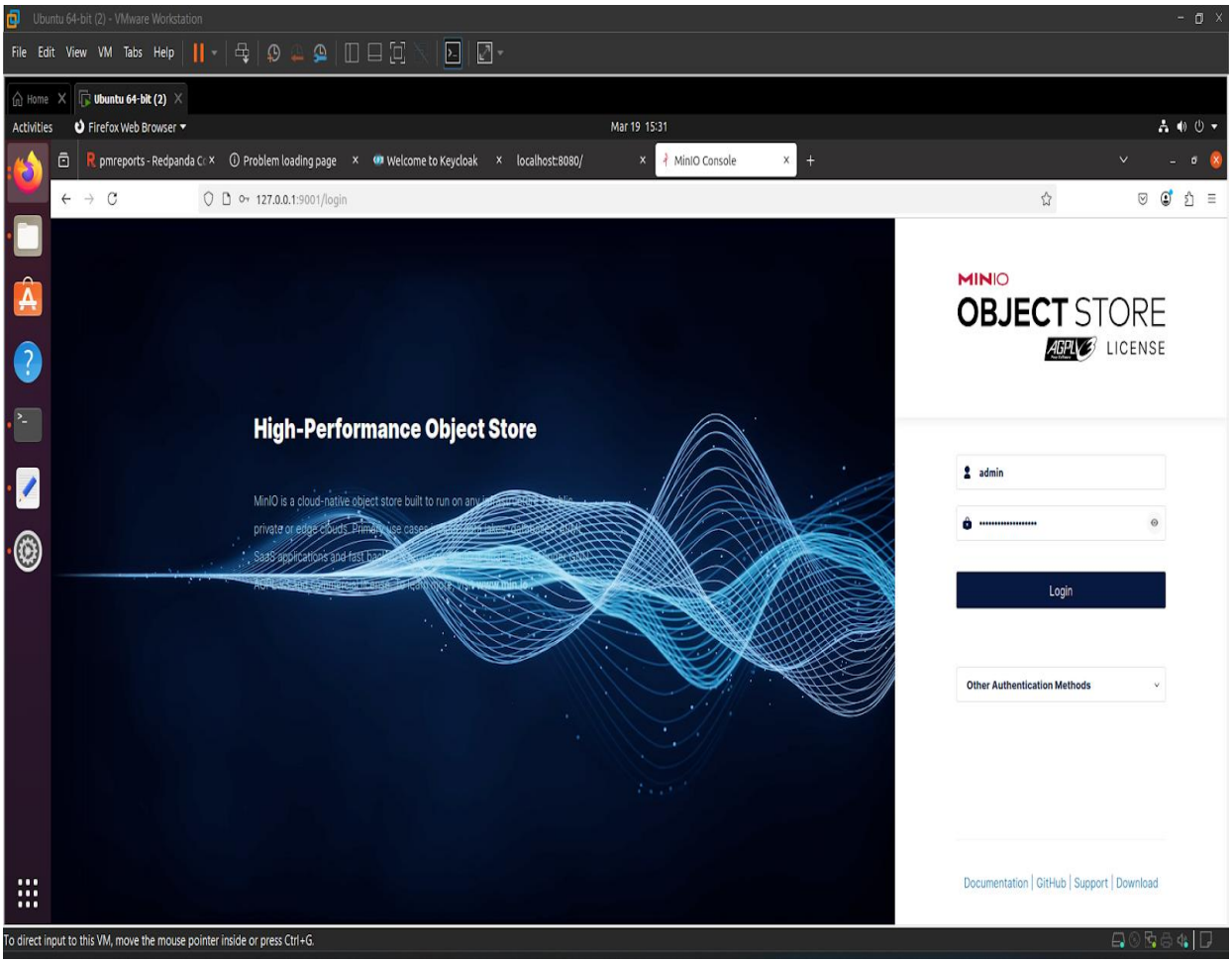


fig. 4-26.minio web interface

- **InfluxDB Browser:**

  Nodeport: 31812

  Credentials: User: admin, Password: mySuP3rS3cr3tT0keN

  Purpose:Visualizes time-series PM data.

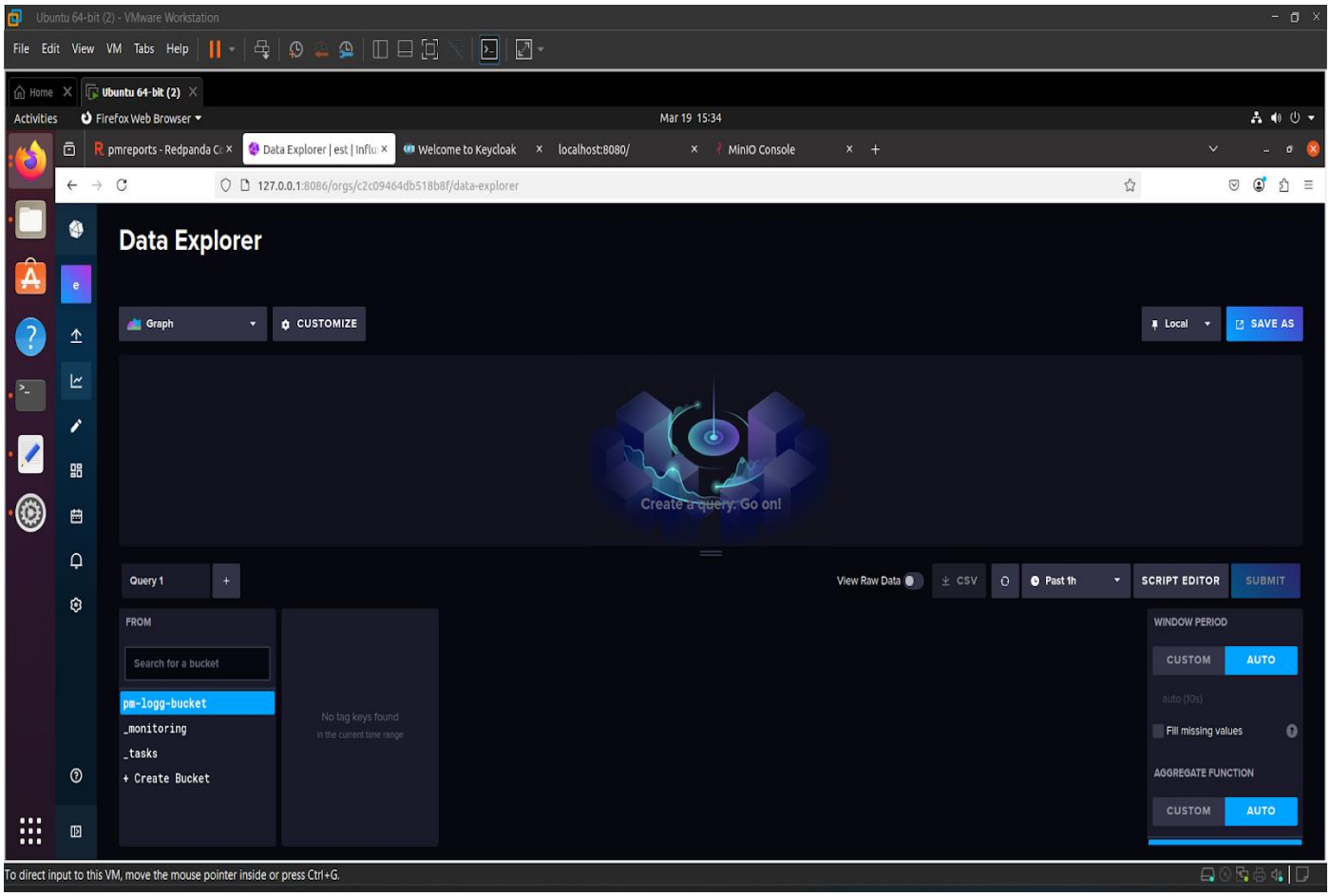


fig. 4-27.influx db browser

For verifying that all components are running in the kubernetes environment:

>kubectl get pods -A

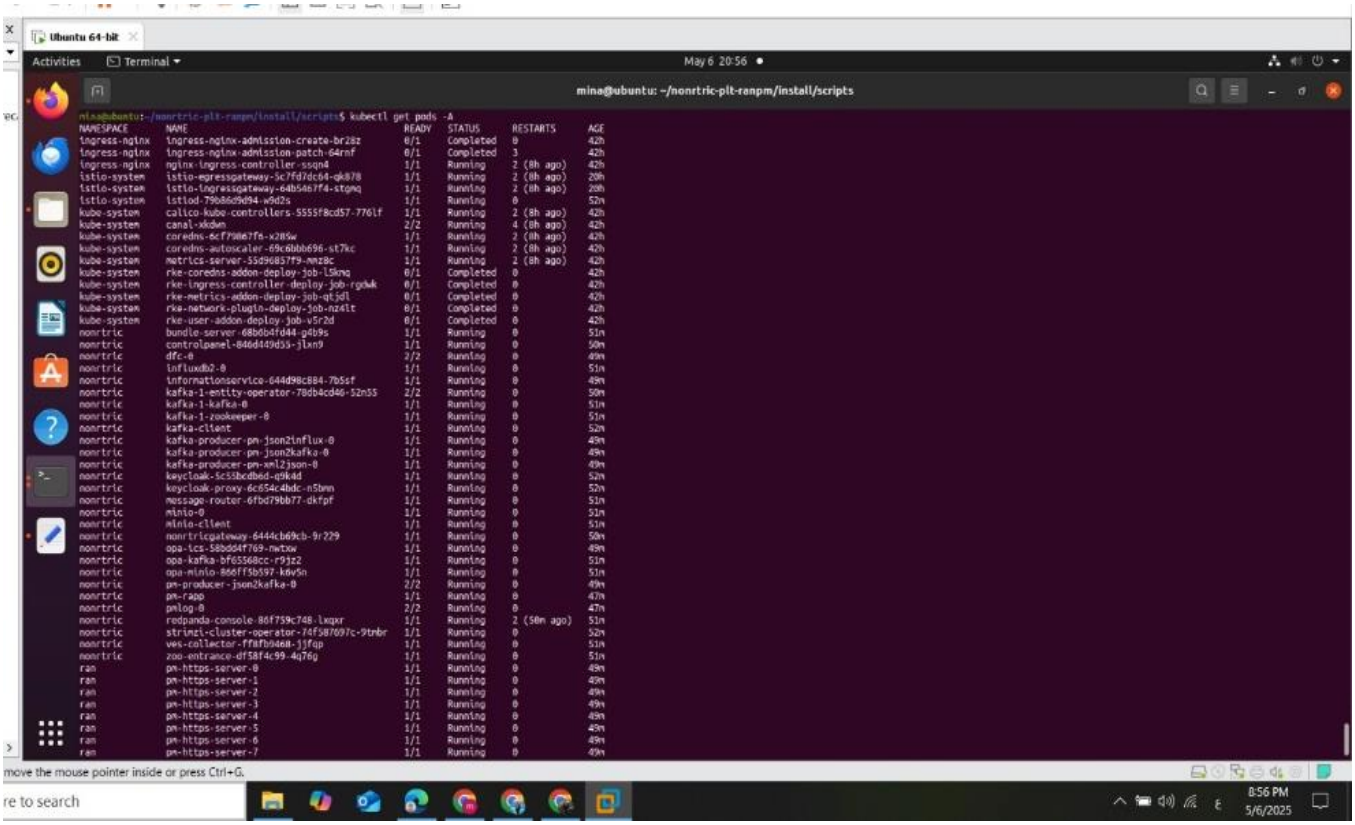


fig. 4-28. verifying Kubernetes cluster

4.1.3.18 Pushing custom data to Kafka and viewing in influx DB

After the initial installation, efforts were made to extend the deployment by pushing custom data to Kafka, beyond the provided dummy data, and visualizing it in InfluxDB to enhance performance monitoring.

Steps:

1. **Navigate to the Installer Directory: Access the installation directory.**
   cd ~/nonrtric-plt-ranpm/install
2. **Execute the Data Push Script: Use the push-genfiles-to-file-ready-topic.sh script to generate and push custom data to the Kafka topic file-ready. The command syntax is:**
   push-genfiles-to-file-ready-topic.sh <node-count> <num-of-events> <node-name-base> <file-extension> <num-servers>
   The specific command used was:
   ./push-genfiles-to-file-ready-topic.sh 20 5 GNODEB xml.gz https 10
   - **Parameters:**
     - **20: Number of nodes.**
     - **5: Number of events per node.**
     - **GNODEB: Base name for nodes.**
     - **xml.gz: File extension for compressed XML data.**
     - **https: Protocol for data transfer.**
     - **10: Number of servers.**
3. **Verify Data in Redpanda: Access the Redpanda Console (nodeport 31767) to confirm the data is available in the file-ready topic.**
4. **Verify Data in InfluxDB: Use the InfluxDB Browser (nodeport 31812) to visualize the ingested data, ensuring it is stored and accessible for analysis.**

Significance: This step demonstrates the ability to integrate custom PM data into the RANPM ecosystem, enhancing real-time monitoring and supporting AI-driven energy

optimization strategies like RF Channel Reconfiguration by providing actionable performance metrics.

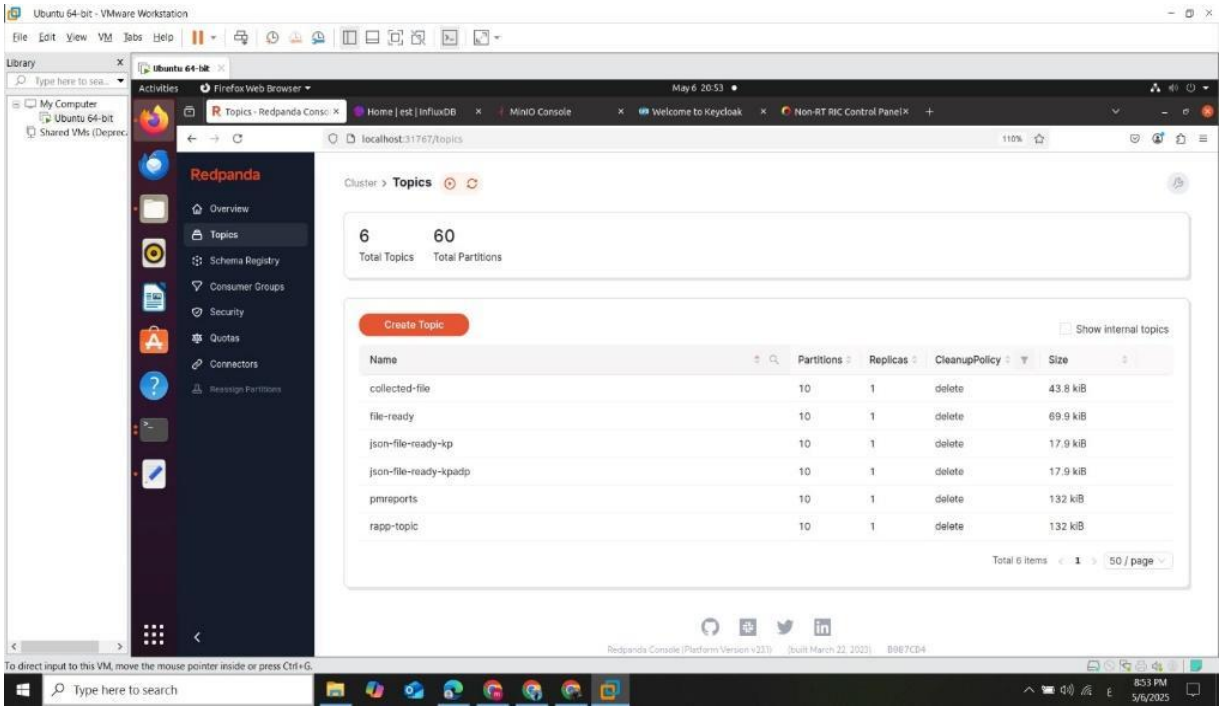
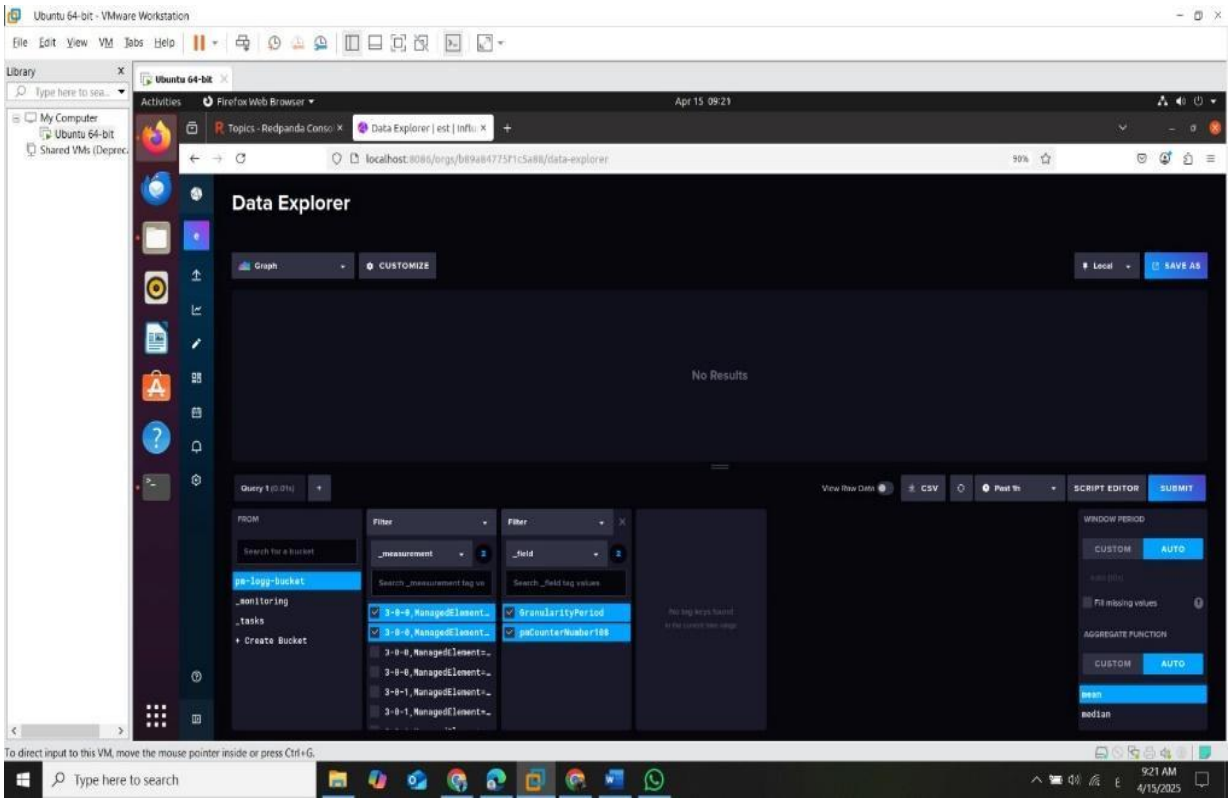


fig. 4-29.Redpanda with pushing data



UE cells view:

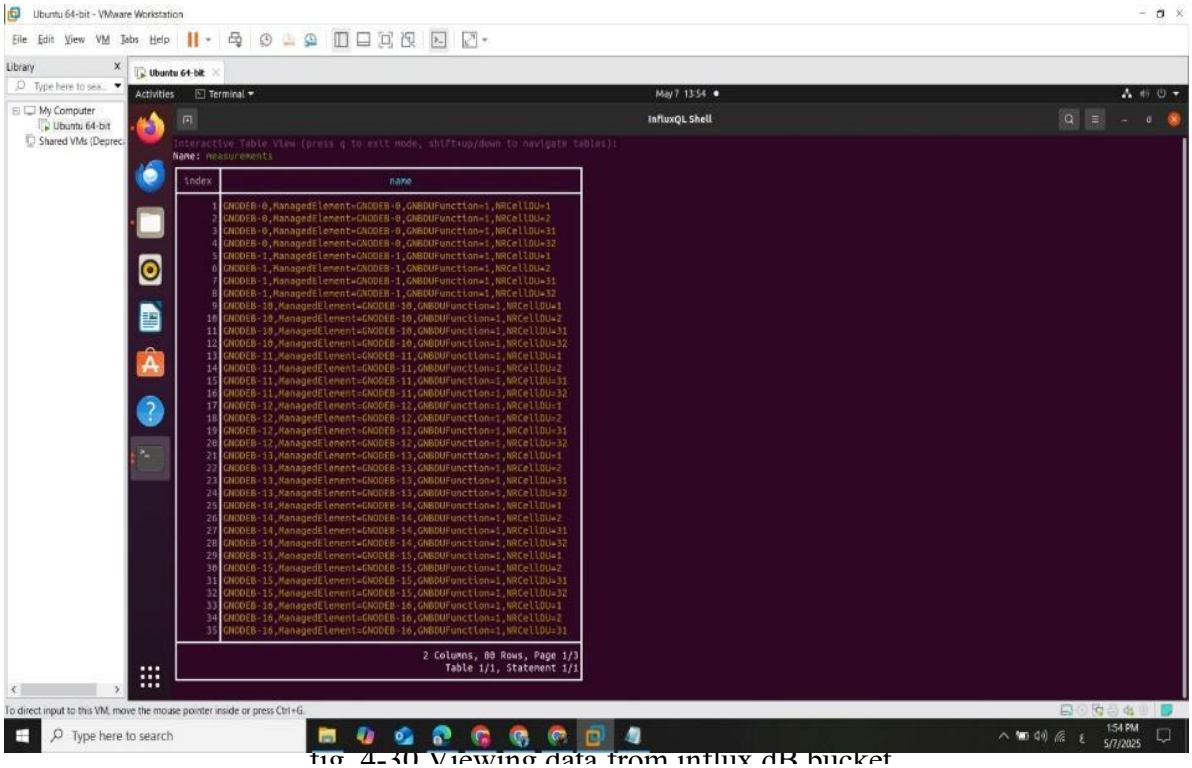


fig. 4-30 Viewing data from influx dB bucket

fig. 4-33. UE cell view

fig. 4-32.Viewing data from influx shell

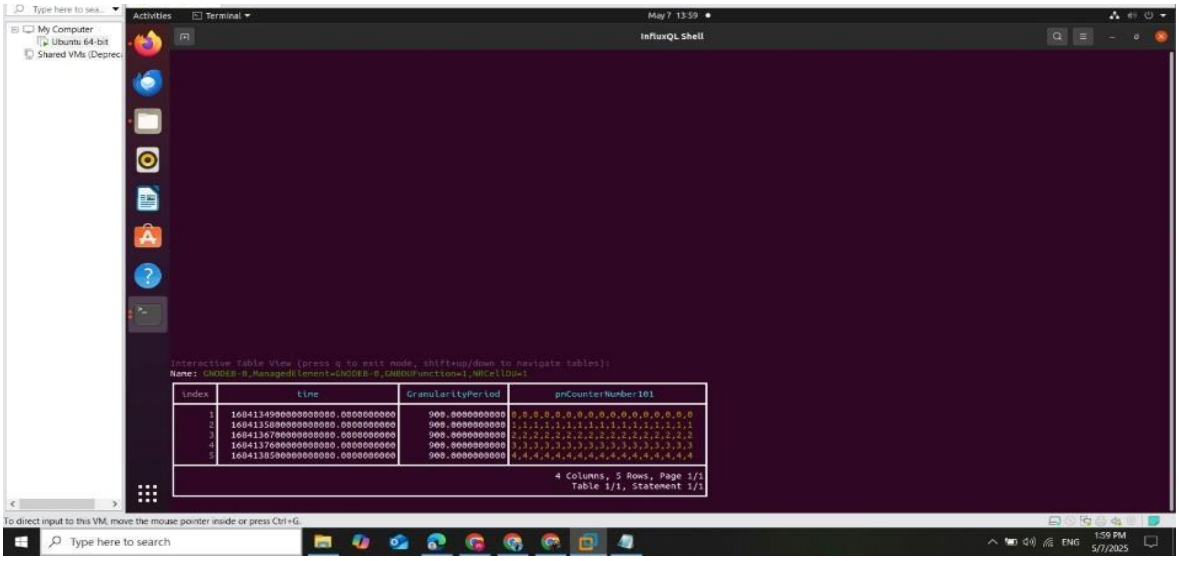


fig. 4-31. Ue cell view

4.1.3.20 Installing additional RANPM Components

Additional scripts deploy supplementary functionalities to enhance RANPM's capabilities:

- **install-pm-log.sh: Installs the producer for Influx DB, enabling storage of PM data in time-series format.**
  ./install-pm-log.sh
- **install-pm-influx-job.sh: Sets up an alternative job to produce data stored in Influx DB, supporting custom data ingestion scenarios.**
  ./install-pm-influx-job.sh
- **install-pm-Rapp.sh: Installs a RAN application (Rapp) that subscribes to and prints received PM data, facilitating debugging and monitoring.**
  ./install-pm-Rapp.sh
- **Significance: These scripts extend RANPM's functionality, supporting flexible data storage and real-time monitoring, critical for AI-driven energy optimization in the O-RAN ecosystem.**

4.2 RAPP Manager

Rapp Manager overview:

The Rapp Manager is a crucial component within the Non-Real-Time RAN Intelligent Controller (Non-RT RIC), responsible for managing the lifecycle of Rapps (RAN Applications). It facilitates the deployment, configuration, execution, and monitoring of Rapps, ensuring their seamless operation within the Open RAN ecosystem.

4.2.1 components

Architecture:

1. NONRTRIC Layer

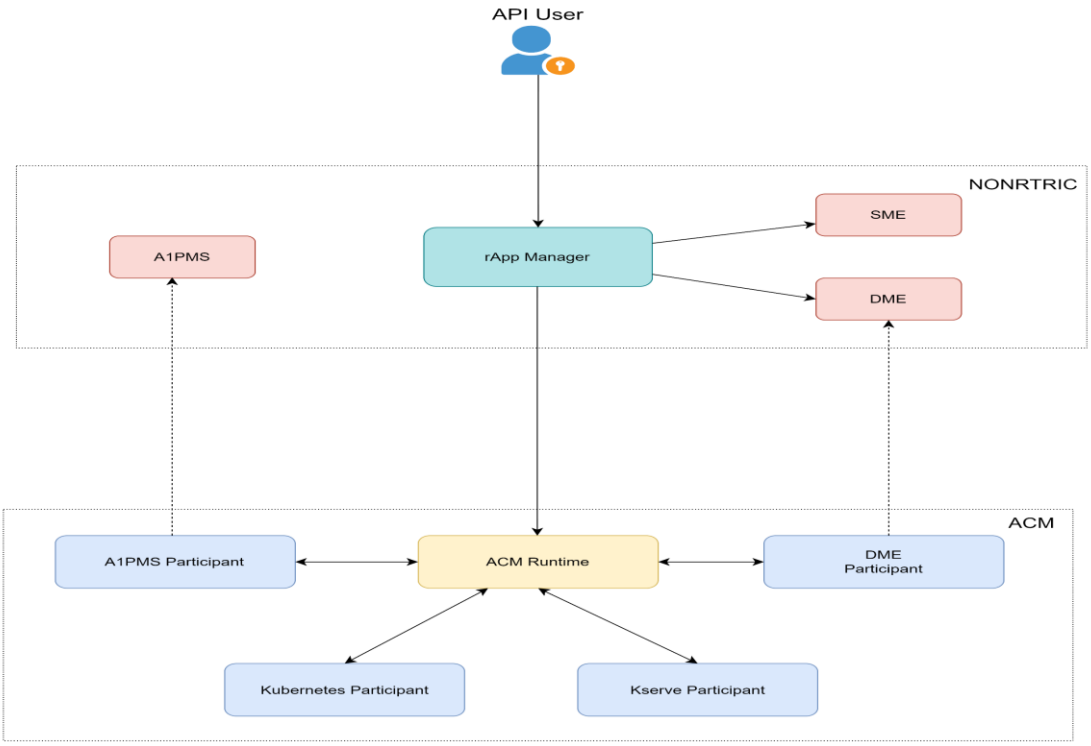


fig. 4-34.Rapp manager architecture

This is the Non-Real-Time RIC (Radio Intelligent Controller) part, where high-level management and orchestration of Rapps (RIC applications) occurs.

Rapp Manager

- The central component responsible for managing the lifecycle of Rapps
- It is the bridge between user input and lower-level runtime actions.

External Modules:

A1PMS: A1 Policy Management Service.

The A1PMS participant receives A1 policy service information from the CLAMP runtime and creates the A1 policy service in A1PMS. The participant acts as a wRapper around the A1PMS and creates the policy service It supports the message Broker Kafka. When an automation composition is initialized, the A1PMS participant starts a A1PMS Automation Composition element for the automation composition. It reads

theconfiguration information sent from the Automation Composition Runtime and runs a REST client to talk to the A1PMS endpoint receiving the REST request

SME: Service Management Exposure.

- Service Manager implements the R1 interface by providing a wRapper for R1 APIs that are produced and/or consumed by Rapps. Service Manager also provides a wRapper for the Kong API Gateway.
- If an Rapp app needs access to a published service, it needs to go through Service Manager. It cannot access the published service directly.

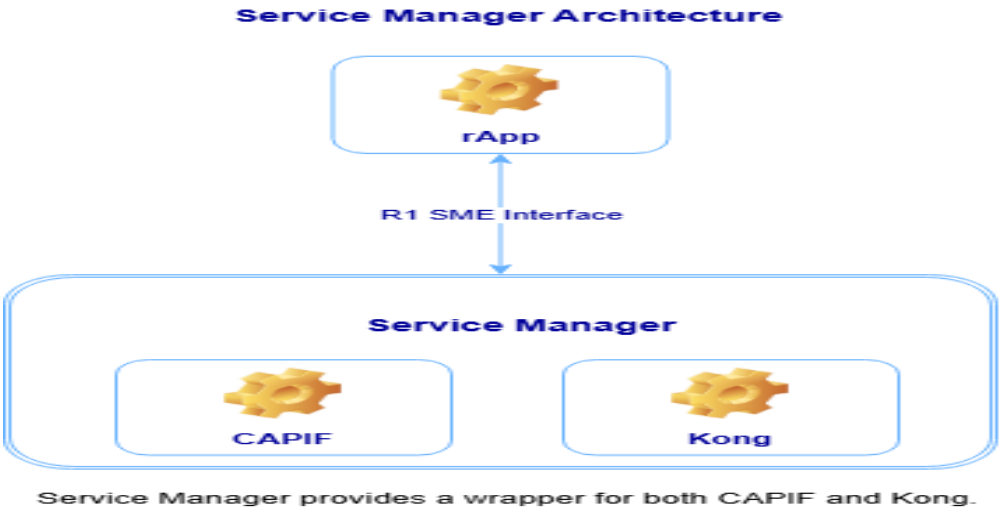


fig. 4-35. service manager and the R1 Api

Service Manager and the R1 API

- CAPIF stands for Common API Framework, and it was developed by 3GPP to enable a unified Northbound API framework across 3GPP network functions, and to ensure that there is a single and harmonized approach for API development. CAPIF (Common API Framework) is defined in 3GPP TS 29.222. Among CAPIF's key features are the following.
- Register/deregister APIs
- Publishing Service APIs
- Onboarding/offboarding API invoker
- Discovery APIs
- CAPIF events subscription/notification

- Entity authentication/authorization
- Support for 3rd party domains i.e., allow 3rd party API providers to leverage the CAPIF framework
- Support interconnection between two CAPIF providers
- The Service Manager wraps CAPIF to provide the following Northbound service APIs.
- Register/deregister APIs
- Publishing Service APIs
- Onboarding/offboarding API invoker
- Discovery APIs
- If you only need the above APIs, then the Service Manager is a plugin-in replacement for CAPIF.

Generation of OpenAPI Code

- The R1 APIs are generated from the OpenAPI specifications provided by 3GPP.
- The generate.sh script downloads the specifications from 3GPP, fixes them and then generates the northbound APIs.
- The specifications are downloaded from https://www.3gpp.org/ftp/Specs/archive/29_series.
- CAPIF needs the server implementation while Service Manager needs both server and client implementations. Service Manager needs the client to access CAPIF and the server to provide access to the Rapps.

Service Manager Integration with Kong

- Service Manager is a Go implementation of a service that wraps both CAPIF and Kong.

- When publishing a service through Service Manager, we create a Kong service and Kong route.
- The JSON element that we return in the response body is updated to point to the Kong data plane. Therefore, the API interface that we return from Service Discovery has the Kong host and port, and not the original service's host and port.
- The Rapp can only access the Published function through Service Manager. It cannot access the Published function directly.
- We use Kong as a reverse proxy. Instead of calling the Publishing service directly, our Invoker's API request is proxied through Kong. This gives us the advantages of using a proxied service, such as providing caching and load balancing.
- When service invocations are routed through Kong, in an R1 SME scenario where services are selectively exposed to Rapps, the Kong API gateway can be used to enforce R1 SME exposure and access.
- Service exposure is achieved by restricting Rapps to have access only to the Kong API gateway, and so only services exposed to the Rapps through the API gateway can be accessed.

DME: Data Management Exposure.

The DME (Information Coordination Service (ICS)) is a generic service that maintains data subscriptions. Its main purpose is to decouple data consumers and data producers in a multi-vendor environment. A data consumer does not need to know anything about the producers of the data.

It is integrated with Rapp manager to enable the Rapp to produce/consume specific type of data (Information Type in DME terms).

2. ACM Layer (Bottom Section)

This is the App Configuration Management runtime environment, which executes and manages the actual deployment/configuration of Rapps.

ACM Runtime

- The core execution engine that handles the actual orchestration and deployment of Rapps.
- Receives instructions from the Rapp Manager.

Participants:

Participants are plugins or modules that extend ACM's capabilities to manage different environments.

A1PMS Participant:
Interface to interact with the A1PMS service above.
DME Participant:
Interface to interact with the DME module in NONRTRIC.
Kubernetes Participant:
Enables ACM to deploy and manage Rapps on Kubernetes clusters.
KServe Participant:
Enables ACM to handle Rapps that are ML models deployed using KServe, a Kubernetes-based model server.

Interaction Flow

1. User initiates a request to manage an Rapp (e.g., deploy or configure it).
2. Rapp Manager in NONRTRIC processes the request.
3. It consults/updates the SME, DME, and A1PMS components as needed.
4. It forwards execution instructions to the ACM Runtime.
5. ACM Runtime uses its Participants to:
    - Deploy workloads to Kubernetes.
    - Manage ML models via KServe.
    - Sync with policy (A1PMS) or data (DME) services.

4.2.2 Implementation:

Software Installation and Deployment

Install with Helm

Helm charts and an example recipe are provided in the it/dep repo, under "nonrtric". By modifying the variables named "installXXX" in the beginning of the example recipe file, which components that will be installed can be controlled. Then the components can be installed and started by running the following command:

bin/deploy-nonrtric -f nonrtric/RECIPE_EXAMPLE/example_recipe.yaml

Install with dependent components

The scripts for the deployments of Rapp Manager and its dependent components are available in *Rappmanager/scripts/install* directory.

ACM components should be configured with couple of other components for the participants to work.

In case some of the installation is already setup or not set by the installation scripts, the below environment variables can be used to set the configurations ACM through installation scripts.

Variable Name	Description	Default Value
CHART_REPO_GET_URI	URI to get the charts. It will be used by Kubernetes participant and sample Rapp generator	http://IP_ADDRESS:8879/charts IP_ADDRESS: IP of the host in which the installation scripts are running.
CHART_REPO_POST_URI	URI to upload the charts. It will be used by sample Rapp generator	http://IP_ADDRESS:8879/charts/api/charts IP_ADDRESS: IP of the host in which the installation scripts are running.
A1PMS_HOST	Address of the A1PMS. It will be accessed from A1PMS participant.	http://policymanagementservice.nonrtric:9080

All components can be installed using the command below,

./install-all.sh

Individual components can be installed using the commands below,

To install the tools required for other installer scripts.

./install-base.sh

To install the ACM, and it's related components.

./install-acm.sh

To install the Kserve, and it's related components.

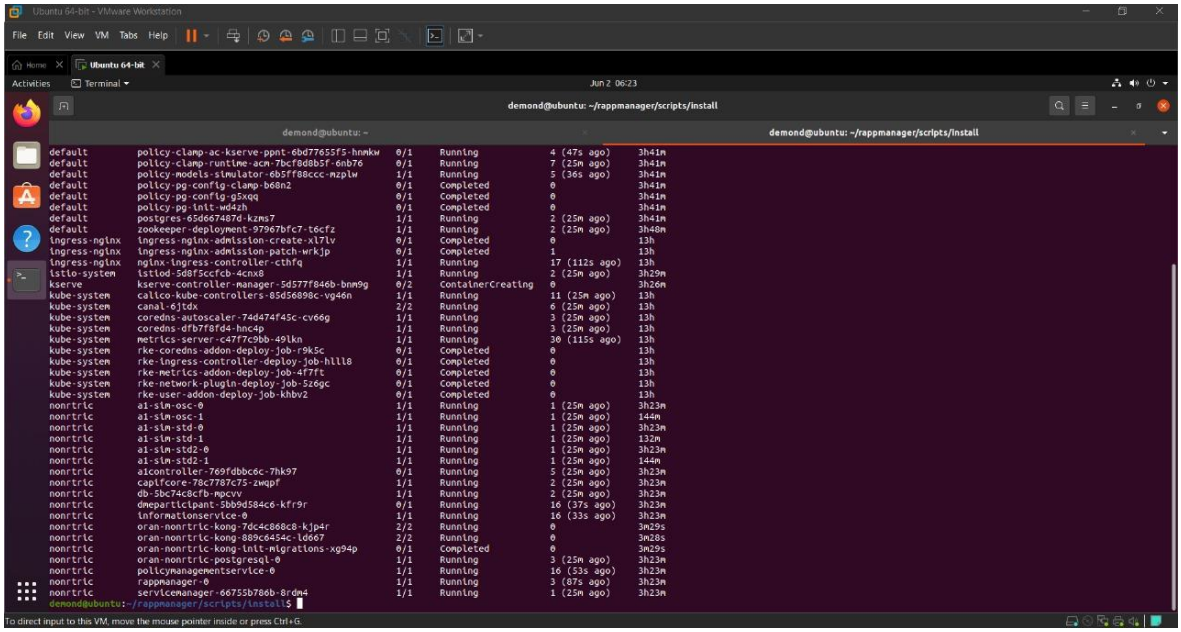
./install-kserve.sh

To installs the NONRTRIC components.

./install-nonrtric.sh

After that ensure all pods are running by :

Kubectl get pods -A



Role of Postman in Project Completion

Postman was employed as an integral tool to finalize the Rapp Manager's API interactions, with the following key roles:

- **API Refinement and Testing:** Postman facilitated the validation of API endpoints, ensuring that onboarding, priming, instantiation, and deployment processes were fully functional within the existing Kubernetes setup.
- **Workflow Standardization:** Collections in Postman organized API requests into a repeatable sequence, ensuring consistency in completing the Rapp lifecycle.

- **Configuration Adaptation:** Environment variables in Postman adapted to the cluster's dynamic settings, such as the Rapp Manager's IP address, enhancing compatibility with the established infrastructure.
- **Documentation Enhancement:** Postman's documentation capabilities provided a comprehensive record of API workflows, supporting the project's final integration and future scalability.

Implementation Details

The implementation of Postman as a complementary tool involved the following steps to complete the Rapp Manager project:

1. **Collection Configuration:** A Postman collection was developed to manage the API workflow, including:
    - **Onboarding Rapps:** Uploading CSAR files (e.g., Rapp-sample-ics-producer.csar) to the Rapp Manager.
    - **Priming Rapps:** Configuring Rapps for instantiation within the existing cluster.
    - **Creating Instances:** Generating Rapp instances in the nonrtric namespace.
    - **Deploying Instances:** Finalizing deployment to the Kubernetes environment.
2. **Environment Setup:** Variables such as the Rapp Manager's service IP (retrieved via kubectl get svc -n nonrtric) were configured to align with the current cluster setup, ensuring seamless integration.
3. **Execution Order:** The collection enforced a logical sequence of API calls, validated against the O-RAN SC specifications, to complete the Rapp deployment process.
4. **Testing and Finalization:** Each request was tested to confirm successful responses (e.g., 200 OK), with Postman's assertions verifying payload integrity, ensuring the project's objectives were met.
5. **Automation Support:** Pre-request scripts in Postman automated data extraction (e.g., instance IDs), streamlining the final deployment stages.

Example Workflow

The following table summarizes the key API requests in the Postman collection used to complete Rapp management:

API Request	Method	Endpoint	Purpose
Onboard Rapp	POST	/onboard	Upload CSAR file to Rapp Manager
Prime Rapp	POST	/prime	Prepare Rapp for instantiation
Create Instance	POST	/instances	Create an Rapp instance
Deploy Instance	POST	/deploy	Finalize deployment to Kubernetes
Start Producer/Consumer	GET	/startProducer/mytopic or /startConsumer/mytopic	Initiate Rapp functionality

Benefits of Using Postman

The integration of Postman significantly enhanced the completion of the Rapp Manager project:

- **Efficient Finalization:** Postman's real-time testing capabilities expedited the validation of API endpoints, ensuring timely project completion.
- **Error Minimization:** Structured collections reduced configuration mistakes, guaranteeing reliable Rapp deployment.
- **Integration Support:** Documentation and environment variables facilitated smooth integration with the existing Kubernetes setup.
- **Debugging Efficiency:** Detailed logs and assertions enabled rapid resolution of API issues, supporting the project's final stages.
- **Future Readiness:** The documented workflows provided a foundation for future Rapp enhancements.

Challenges and Mitigations

Challenges encountered during Postman's use were addressed as follows:

- **Dynamic IP Management:** Changes in the Rapp Manager's IP were mitigated by scripting IP retrieval and updating environment variables.
- **Complex Payload Handling:** Postman's file upload and JSON editor simplified managing CSAR file payloads.
- **Team Familiarity:** Initial training and shared collections resolved the learning curve for new users.

Impact on Project Completion

Postman's role was instrumental in finalizing the Rapp Manager project by:

- **Streamlining Processes:** Automated workflows accelerated the deployment of ICS Producer and Consumer Rapps.
- **Ensuring Stability:** Rigorous testing minimized runtime errors, validating the project's success.
- **Supporting Scalability:** The reusable collection enabled future Rapp deployments with minimal adjustments.
- **Enhancing Documentation:** API documentation served as a resource for project handover and maintenance.

Significance of Rapp from postman:

Performing GET and POST commands of sample Rapp with the terminal and view it from postman GUI

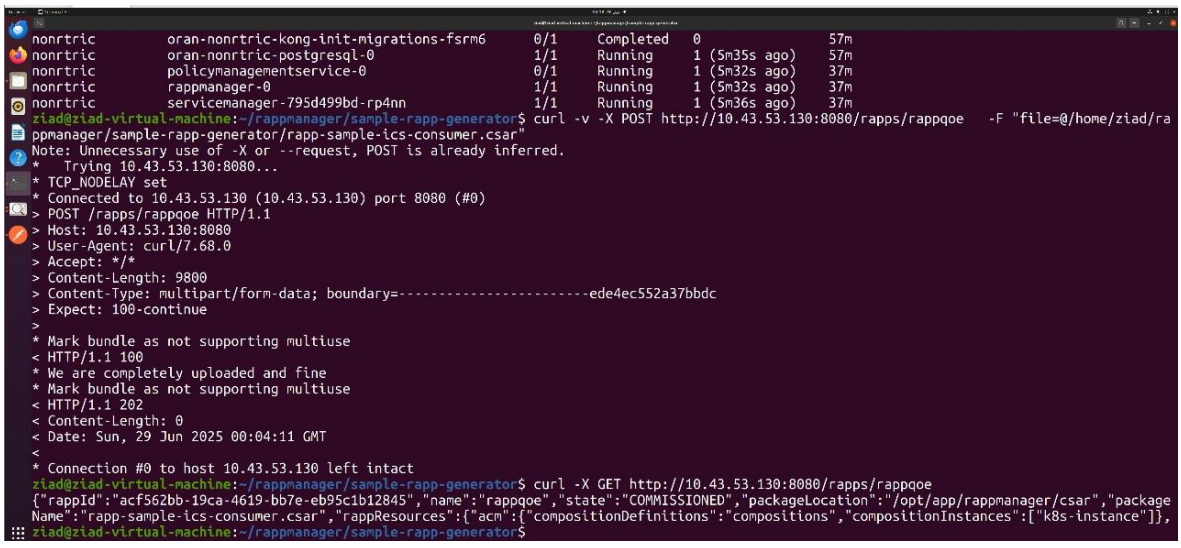


fig. 4-36.GET&POST for Rapp

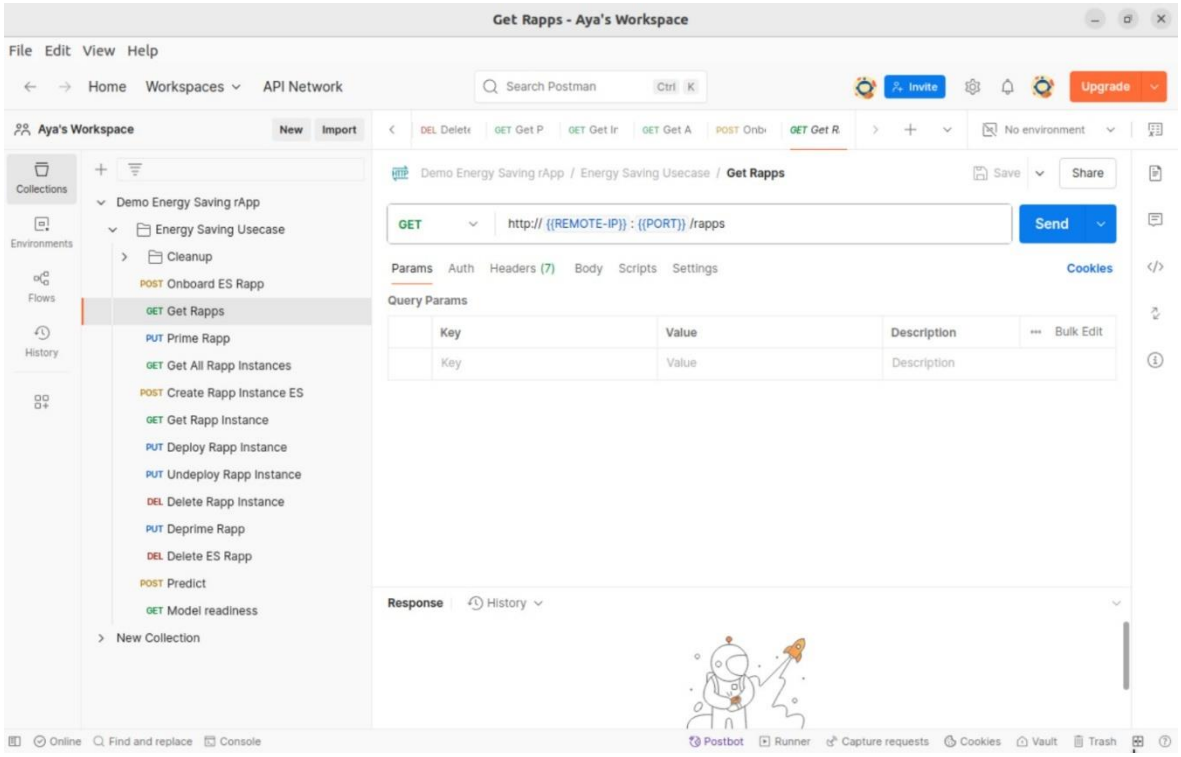


fig. 4-37.Postman GET Request for Rapp Onboarding

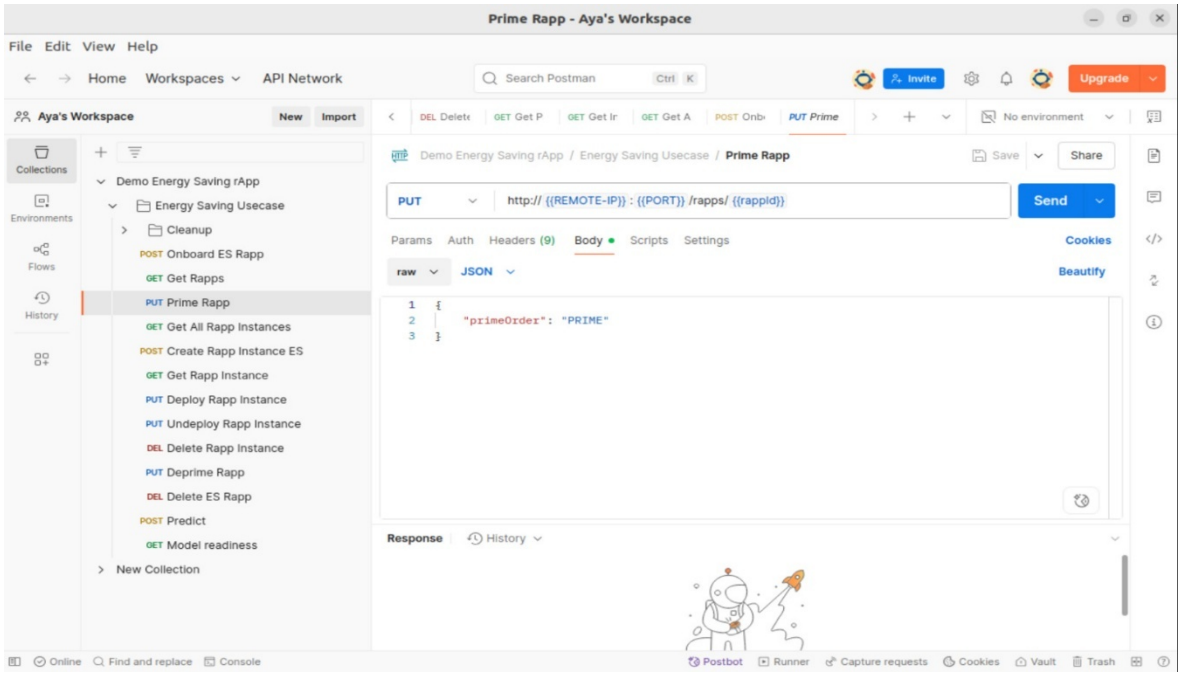


fig. 4-38.Priming the Rapp via PUT with primeOrder "PRIME".

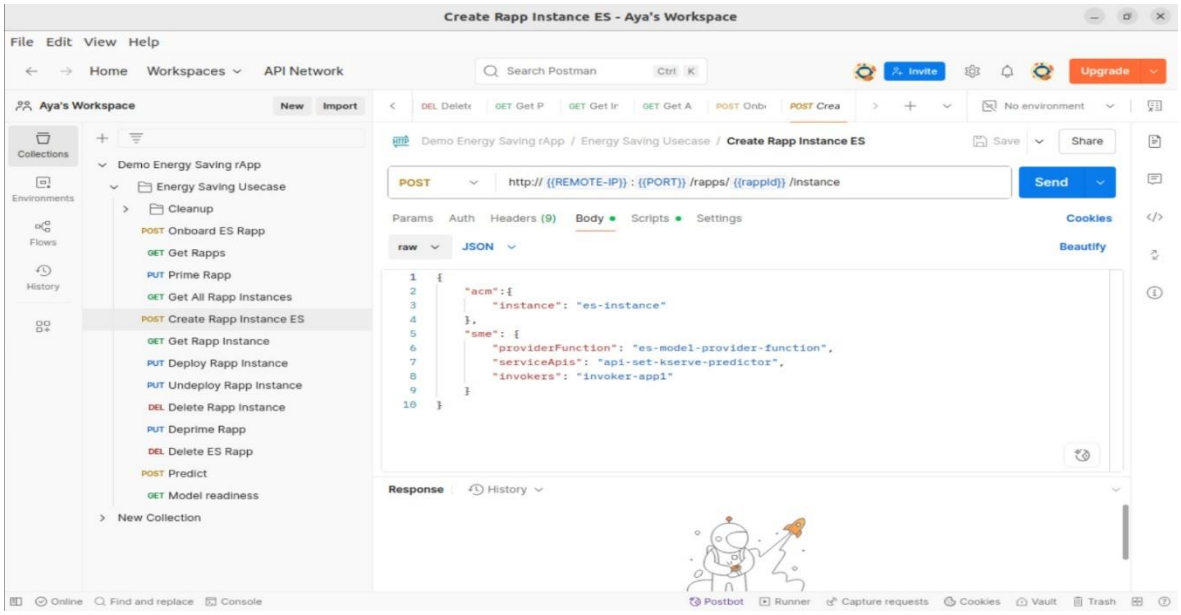
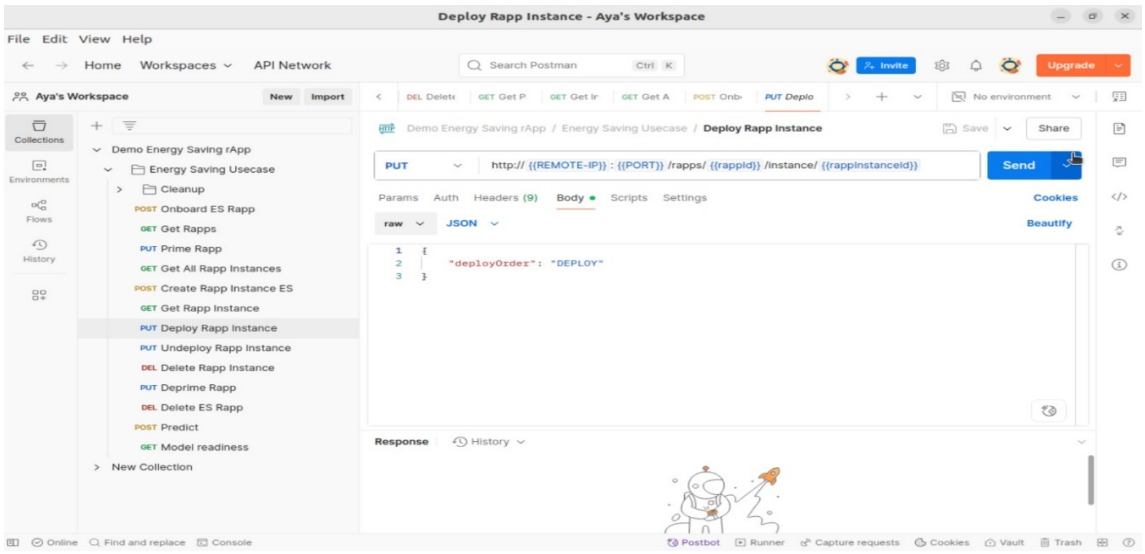
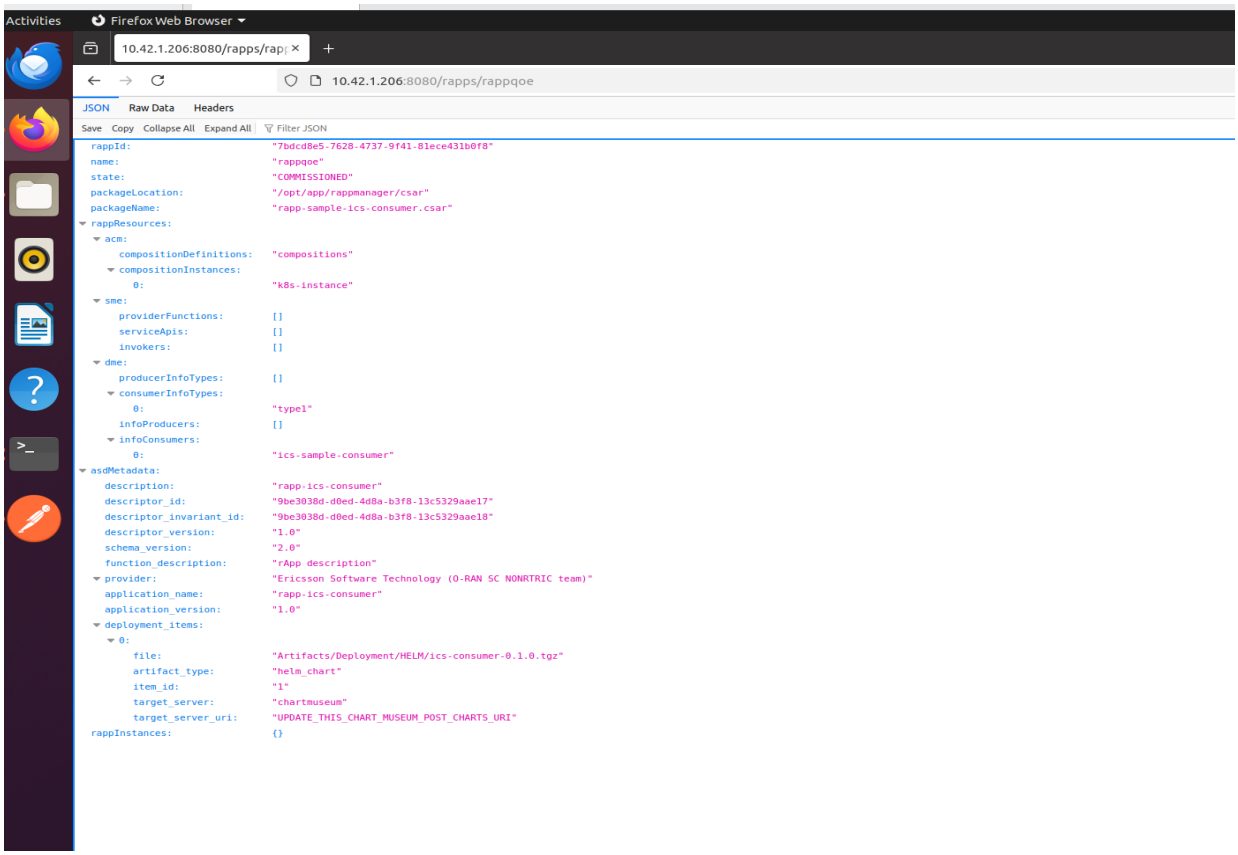


fig. 4-39. Creating an Rapp instance for the Energy Saving use case via POST request in Postman



.

fig. 4-40.Deploying an Energy Saving Rapp via PUT request in Postman.



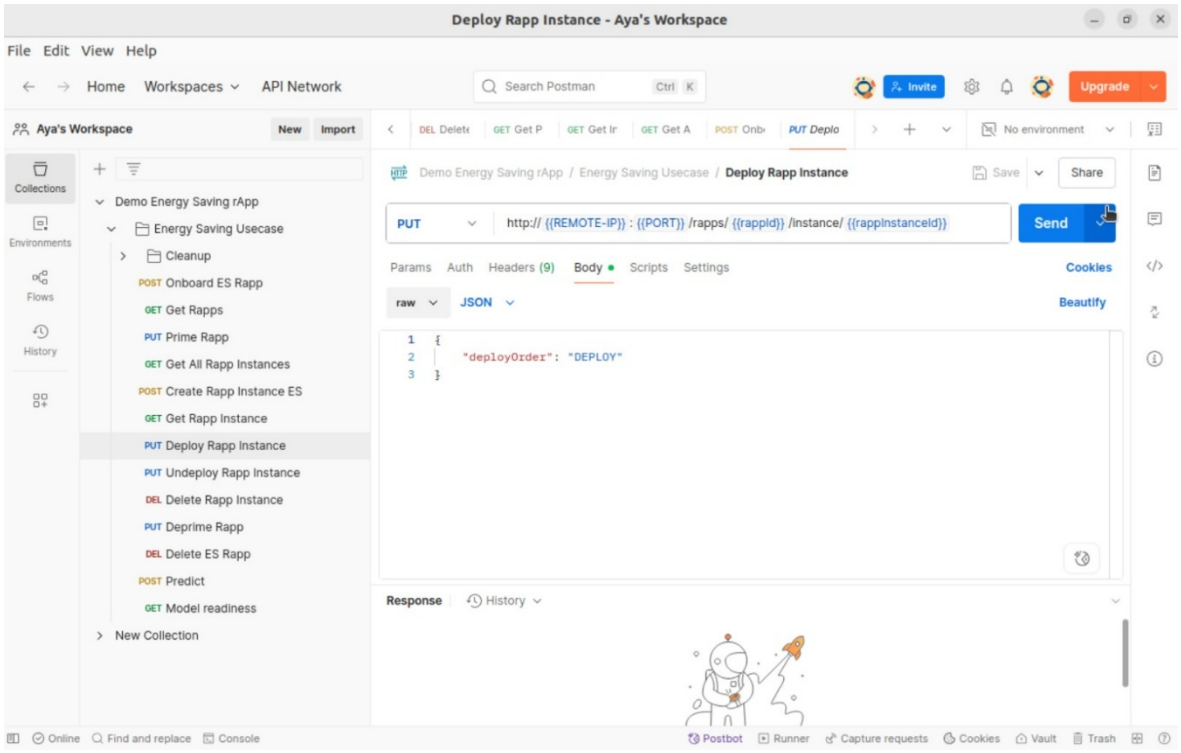


fig. 4-41.Viewing detailed Rapp descriptor metadata via browser API endpoint

Conclusion

Postman's integration as a complementary tool was vital in completing the Rapp Manager project, refining API interactions within an existing Kubernetes environment. Its automation, testing, and documentation capabilities ensured the successful deployment of Rapps, aligning with the project's goals. This approach not only finalized the current implementation but also established a robust framework for future enhancements, underscoring Postman's value in completing complex, API-driven projects.

4.3 RAPP

4.3.1 what is Rapp?

In ORAN architecture, Rapps (RAN Intelligent Controller applications) are software programs that run on the Non-Real Time RIC (Non-RT RIC). This Non-RT RIC is a key component in the O-RAN architecture responsible for handling network management and optimization tasks that don't require immediate, real-time responses. Rapps runs on the non-RT RIC and communicating via the R1 interface and enable intelligent, policy-driven control and optimization of the RAN on a timescale suitable for non-real-time operations. The R1 interface plays a crucial role here as it serves as the communication pathway between the Rapps and the underlying non-RT RIC framework. It enables Rapps to access necessary data, interact with other components, and ultimately contribute to the overall service management and orchestration of the network.

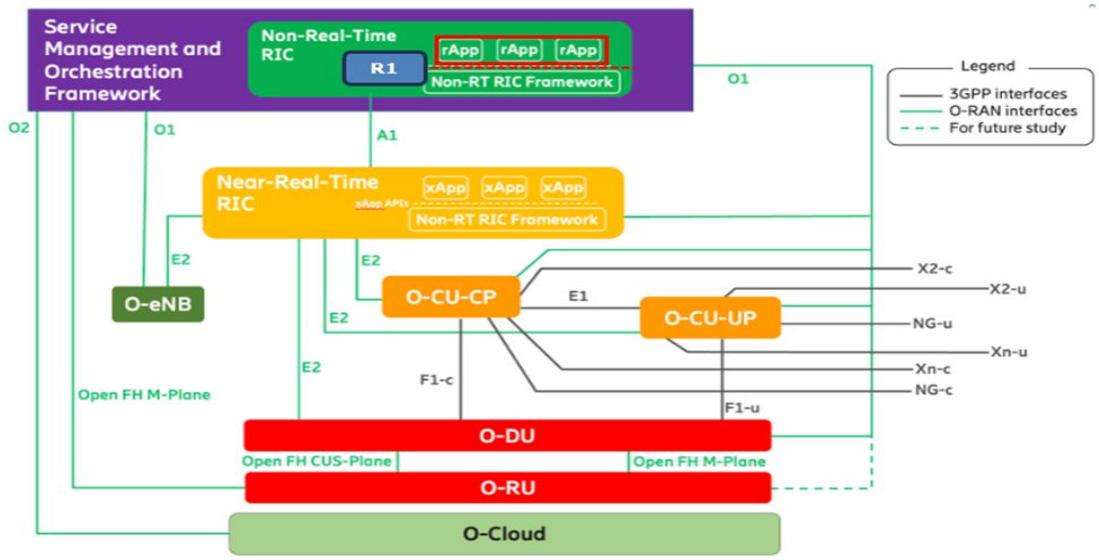


fig. 4-42.open ran architecture

4.3.2 components

Key Features of Rapps:

• Non-Real-Time Optimization: Rapps are used for non-real-time network optimization tasks, which typically have a latency requirement of more than 1 second. This includes functionalities like network planning, resource optimization, and load balancing.

• Control and Management: They interact with the RAN components through the Non-RT RIC via standardized interfaces and play a crucial role in managing the RAN elements over a longer time scale.

• Policy Guidance and Optimization: Rapps provide policy guidance, configuration, and optimization decisions to the near-Real-Time RIC (near-RT RIC), which further interacts with the RAN components for real-time control.

• Integration with AI/ML Models: Rapps can host AI/ML models for tasks such as predictive analysis, anomaly detection, and traffic forecasting, which require data aggregation and longer processing times.

• Support for Network Slicing: Rapps can be used to manage and optimize network slices, ensuring each slice gets the required resources and meets the quality of service (QoS) parameters.

Common Use Cases of Rapps:

Rapps represents a cornerstone of intelligent network management within the O-RAN architecture. Operating on the Non-Real Time RIC, Rapps are specifically designed to handle tasks that don't demand immediate, real-time responses. Leveraging their capabilities in data analysis, policy implementation, and automation, Rapps empower network operators to optimize network performance, enhance energy efficiency, and enable a multitude of innovative use cases, contributing significantly to the evolution of 5G and beyond networks.

Followings are some of common use cases of Rapps :

- Traffic Steering

Rapps can analyze network traffic patterns, congestion levels, and user demands in a non-real-time manner to make intelligent decisions on how to route traffic. This may involve dynamically adjusting routing paths, load balancing across different cells, or even steering users to different frequency bands or network slices based on their service requirements. By doing so, Rapps can enhance network performance, reduce congestion, and ensure an optimal quality of experience for users.

- Energy Savings

Rapps can play a crucial role in energy-saving initiatives within the RAN. They can analyze historical data, traffic patterns, and cell load over longer durations to identify opportunities to put certain network elements into sleep modes or adjust their

power levels during periods of low utilization. They can also implement intelligent energy-saving policies based on user behavior and network conditions, contributing to a greener and more sustainable network.

- Anomaly Detection

Rapps can leverage their access to historical data and machine learning capabilities to identify abnormal patterns or behaviors within the network. This may include detecting sudden spikes in traffic, unusual signaling patterns, or performance degradations that could indicate potential issues or security threats.

By detecting these anomalies early on, Rapps can trigger timely actions, such as alarms, diagnostics, or even automated mitigation steps, to prevent service disruptions and ensure network reliability.

R1 : Interface between Rapp and Non-Near RT RIC

The R1 interface is a standardized and consistent communication channel between Non-RT RIC and the Rapps. It abstracts the complexity of the underlying network while providing a set of services and data that the Rapps can leverage to enhance RAN performance, support automated network management, and execute non-real-time control functions.

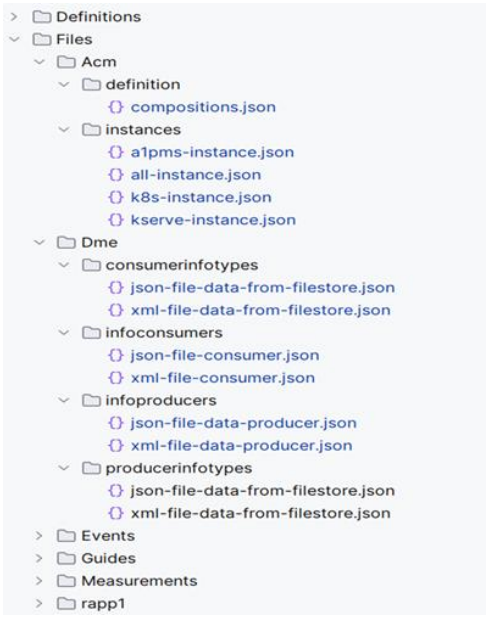
A1 Related Services:

A1 Related Services specifically address the management and orchestration of A1 policies. These policies are instrumental in governing the dynamic interactions and behaviors between the Non-RT RIC and the near-RT RIC . By providing a structured framework for policy creation, modification, and enforcement, A1 Related Services enable intelligent and adaptive control of the RAN, facilitating efficient resource allocation, optimized performance, and support for a wide array of 5G use cases.

• A1 Policy Management: This API enables the creation, retrieval, modification, and deletion of A1 policies. These policies govern the behavior and interactions between the Non-RT RIC and the near-Real-Time RIC, playing a crucial role in intelligent and dynamic control of the RAN.

ASD Package:

fig. 4-43.ASD package



The Application Service Descriptor (ASD) defines how an Rapp is structured and deployed. It contains metadata, dependencies, and configurations required for operatioContents of an ASD Package:

Rapp Configuration: Required parameters and dependencies.

Execution Instructions: Defines how the application should be started and managed.

Connectivity Details: Specifies interactions with other system components

Rapp States:

An Rapp transitions through multiple states during its lifecycle:

1. COMMISSIONED: Rapp get created in this state and once the DEPRIMING is completed

2. PRIMING: This is a transition state. Rapp will be in this state once the PRIMING requested for Rapp

3. PRIMED: Rapp will be in this state once the PRIMING is completed. In this state Rapp instances can be created

4. DEPRIMING: This is a transition state. Rapp will be in this state once the DEPRIMING requested for Rapp

4.3.3 implementation

In this section, we detail the practical implementation of a custom Rapp designed for energy-saving in 5G networks. This Rapp runs on the Non-RT RIC and uses telemetry data to make intelligent decisions about powering on or off RAN cells based on predicted traffic load using a machine learning model.

Module Overview

The Rapp consists of the following main Python modules:

Table 2,Rapp python modules

Module	Description
DATABASE	Connects to InfluxDB, reads historical data.
ASSIST	Handles communication with the ML model (sending data & receiving predictions).
NCMP_CLIENT	Interface for controlling RAN cells — power ON/OFF actions.
PolicyManager	(Optional) Interacts with A1PMS to apply policy-based actions.
Schedule	Used to periodically run the ML inference logic.

Class Initialization:

```
class ESrapp():
    def __init__(self, generate_data_local_db=False):
        self.cell_power_status = {}
        self.db = DATABASE()
        self.assist=ASSIST()
        self.db.connect()

        if generate_data_local_db:
            self.db.generate_synthetic_data()
        else:
            self.db.get_url_from_sme()

        self.threshold = 50
        self.ncmp_client = NCMP_CLIENT()
        self.index = 1
```

fig. 86.Class Initialization:

Explanation:

- cell_power_status: Tracks the ON/OFF state of each cell to avoid repeated actions.
- DATABASE & ASSIST: Core interfaces for data and ML communication.
- If running locally, it can generate fake data for testing.
- NCMP_CLIENT: Used to send control commands to RAN nodes.

Execution Loop

```
    def entry(self):
        schedule.every(10).seconds.do(self.inference)

        while True:
            schedule.run_pending()
```

fig. 87.Execution Loop

Explanation:

- Sets up a loop to run inference every 10 seconds.
- Uses schedule to call self.inference() repeatedly.

ML Inference & Decision Making:

```python
# Send data to ML rApp
def inference(self):
    data = self.db.read_data()
    if data.empty:
        logger.info("No data to process... skipping this iteration of inference.")
        return

    data_mapping = self.mapping(data)
    groups = data_mapping.groupby("CellID")

    for group_name, group_data in groups:
        json_data = self.generate_json_data(group_data)
        logger.info(f"Send data to ML rApp {group_name}: {json_data}")
        status_code, response_text = self.assist.send_request_to_server(json_data)

        if not self.check_and_perform_action(response_text):
            self.control_cell(group_data, group_name, turn_on=True)
        else:
```

fig. 4-44.ML Inference & Decision Making

Explanation:

- Reads KPI data from DB.
- Groups by cell.
- Sends each group's data to an ML model.
- Based on the prediction, turns the cell ON or OFF using control_cell().

Cell Control Logic

```python
#Cell Control Logic
def control_cell(self, group_data, group_name, turn_on):
    cell_id_number = group_data['cellidnumber'].iloc[0]
    action = "on" if turn_on else "off"
    logger.info(f"Turn {action} the cell {group_name}")
    time.sleep(3)

    if cell_id_number not in self.cell_power_status:
        self.cell_power_status[cell_id_number] = "off" if turn_on else "on"

    current_status = self.cell_power_status[cell_id_number]
    if (turn_on and current_status == "on") or (not turn_on and current_status == "off"):
        return

    if turn_on:
        self.ncmp_client.power_on_cell(cell_id_number)
    else:
        self.ncmp_client.power_off_cell(cell_id_number)
```

fig. 4-45.cell control logic

Explanation:

- Performs the power ON/OFF command for a cell.
- Waits 3 seconds before acting (safe delay).
- Uses a local status cache to prevent unnecessary switching.

Data Preparation for ML

```
def generate_json_data(self, data):
    # rrc_conn_mean_values = data["RRC.ConnMean"].tolist()
    drb_ue_thp_ul_values = data["DRB.UEThpUl"].tolist()
    rru_prb_used_ul_values = data["RRU.PrbUsedUl"].tolist()
    pee_avg_power_values = data["PEE.AvgPower"].tolist()
    instances = [
        [
            [drb, rru, pee]
            for drb, rru, pee in zip(
            drb_ue_thp_ul_values,
            rru_prb_used_ul_values,
            pee_avg_power_values
        )
        ]
    ]
    json_data = {"signature_name": "serving_default", "instances": instances}
    logger.debug(f'Generated JSON data: {json_data}')
    return json_data
```

fig. 90.data preparation for ML

Explanation:

- Prepares data in JSON format suitable for the ML model.

Prediction Handler

```
def check_and_perform_action(self, data):
    response_obj = json.loads(data)
    predictions = response_obj.get('predictions')
    if predictions:
        for prediction in predictions:
            if all(pred < 0.04 for pred in prediction):
                return True
            elif all(pred >= 0.04 for pred in prediction):
                return False
    return False
```

fig. 4-46.prediction handler

Explanation:

- Interprets model output.
- If all predictions are below 0.04 → network cell is underutilized → Turn OFF.
- Otherwise → Turn ON.

Summary:

This implementation integrates with:

- InfluxDB for KPI data,
- ML models for traffic prediction,
- NCMP for cell power management,
- And optionally with A1PMS/DME/SME for full O-RAN compliance.
- It shows the power of Rapps in performing intelligent, policy-driven decisions for energy optimization in Open RAN networks.

5. Chapter 5 developer assistant using agentic ai

5.1 intro

Large Language Models (LLMs) have become essential tools for understanding and generating human-like language. However, general-purpose LLMs often struggle in highly technical fields like **Open RAN** and **MIMO**. To address this, our project builds a **custom Retrieval-Augmented Generation (RAG)** system designed to support and enhance understanding in these advanced telecommunications domains.

By integrating tools such as **Llama Index** for document retrieval and **sentence-transformers** for semantic search, the system can provide accurate, context-aware responses based on specialized documentation. We also leverage modern **LLM gateways** like **Open Router** and **Groq Cloud** to access high-performance models through unified APIs.

This RAG-based approach allows us to combine the strengths of LLMs with reliable domain knowledge, making it easier for engineers and researchers to interact with complex Open RAN and MIMO information through natural language.

5.2 objective

- The main objective of this project is to develop an intelligent assistant capable of:
- Enhancing the understanding of technical documentation in the telecommunications field, specifically Open RAN and MIMO
- Enabling natural language interaction with dense and complex materials
- Improving accuracy, efficiency, and reliability in retrieving and interpreting domain-specific knowledge
- Supporting local or cloud-based deployment with privacy-aware configurations
- Ultimately, the project aims to bridge the gap between general LLM capabilities and specialized technical knowledge, making it more accessible and usable through modern retrieval and generation techniques.

5.3 LLM

5.3.1 Definition

Large Language Models (LLMs) are advanced deep learning models designed to understand and generate human language. Built using neural network architectures with billions or even trillions of parameters, LLMs process large-scale text data using **self-supervised learning** techniques. They enable powerful natural language applications such as **chatbots**, **machine translation**, **summarization**, **code generation**, and **conversational AI**.

5.3.2Architecture and Operation of LLMs

Large Language Models (LLMs) are based on transformer architectures that use **self-attention** to understand word relationships. Unlike RNNs or LSTMs, transformers process sequences in parallel, making them faster and more scalable for GPU/TPU training.

Core components include:

- **Embeddings**: Convert tokens to vectors.
- **Positional Encoding**: Adds order info.
- **Encoder-Decoder**: Analyzes input and generates output.
- **Self-Attention** & **Multi-head Attention**: Weigh and combine context.
- **Output Layer**: Predicts next token.

5.3.3 Challenges in Training LLMs

LLMs come with several constraints:

- **High Costs**: Require extensive computational resources.
- **Outdated Training Data**: LLMs are often trained on static snapshots of the internet or datasets collected months or years prior, which limits their awareness of recent events or newly emerging knowledge.
- **Training Time & Energy Use**: Weeks of training can incur large environmental impacts.
- **Data Access**: Legal and ethical concerns around large text datasets.
- **Scaling Limits**: Performance gains decrease beyond certain model sizes..

5.3.4 LLM Gateway

LLM Gateway is a centralized orchestration platform designed to streamline the deployment and management of Large Language Models (LLMs) in production. It provides a unified interface that abstracts the complexity of interacting with individual models, enabling seamless integration between applications and one or more LLMs.

What and why **Ollama?**

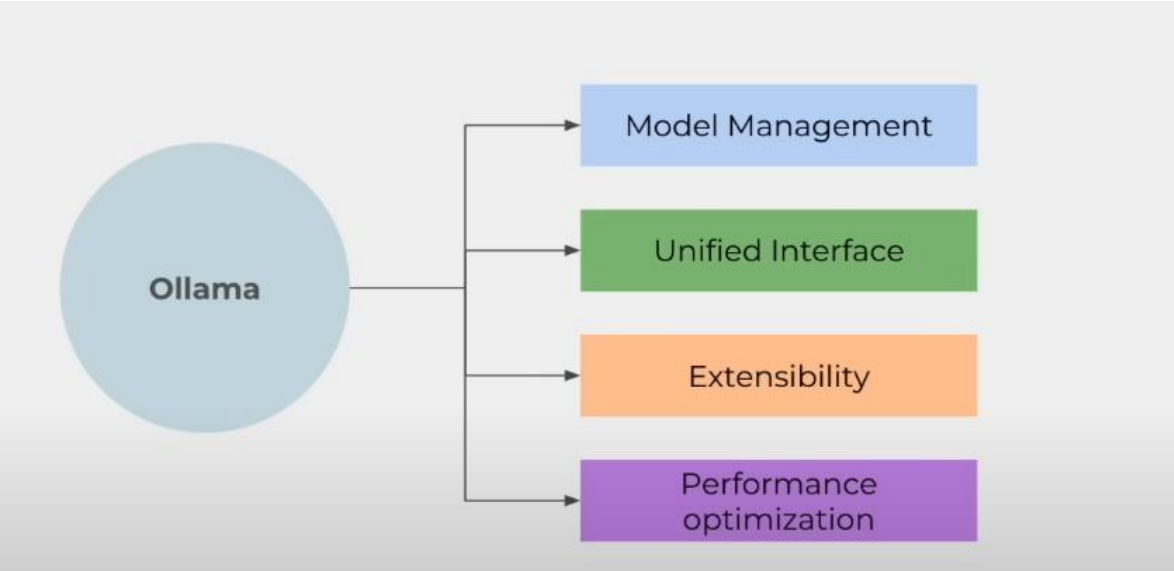


Fig. 0-1- Ollama architecture

Ollama (Omni-Layer Learning Language Acquisition Model) is a machine learning platform designed to run large language models (LLMs) **locally**. It offers a user-friendly interface and seamless integration, enabling individuals and organizations to easily utilize LLMs without relying on cloud-based solutions. Ollama aims to democratize access to advanced natural language processing capabilities across various applications.

The **Ollama** platform offers a curated collection of Large Language Models (LLMs), supporting a wide range of tasks and use cases. These models vary in size and specialization, enabling users to choose the most suitable option for their application needs. Examples include:

- **Phi-3** (3.8B)
- **Mistral** (7B)
- **Code Llama** (7B)

What problem does it solve

1. **Privacy concerns** Running LLMs locally provides a high level of security and protects sensitive data.
2. **Ease of use Setting up** LLMs can be challenging. With Ollama, the process is easy and user-friendly.
3. **Cost efficiency** Eliminates the need for costly cloud-based services, making it more budget-friendly.

5.4 Rag System Architecture

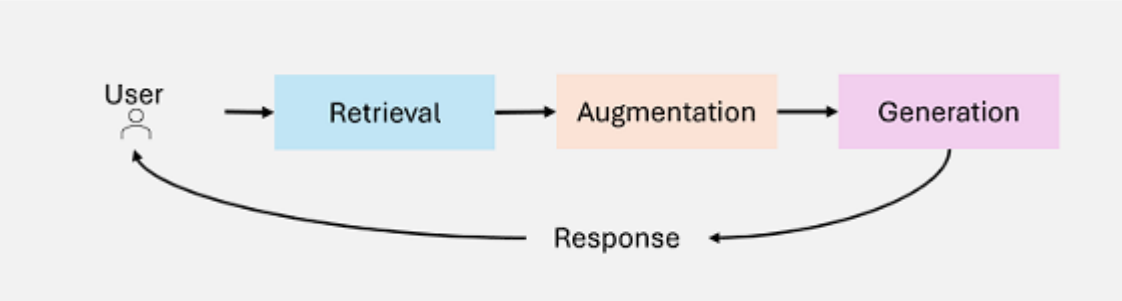


Fig. 0-2. RAG Archeciture

The architecture of RAG systems integrates three primary components.

- Retrieval: Responsible for querying external data sources such as knowledge bases, APIs, or vector databases. Advanced retrievers leverage dense vector search and transformer-based models to improve retrieval precision and semantic relevance.
- Augmentation: Processes retrieved data, extracting and summarizing the most relevant information to align with the query context.
- Generation: Combines retrieved information with the LLM's pre-trained knowledge to generate coherent, contextually appropriate responses.

Evolution of RAG Paradigms:

RAG systems have evolved from basic keyword-based retrieval to advanced, modular frameworks that support contextual accuracy, scalability, and multi-step reasoning. Modern RAG integrates diverse data sources and supports autonomous decision-making, enabling it to handle complex real-world queries more effectively. For achieve all of that will need some main component:

- Knowledge Sources
- LLM Gateway (Ollamh, Open-Router,Groq)
- RAG framework (Llama Index)

5.4.1 Rag Types

5.4.1.1 Naïve RAG

Naïve Retrieval-Augmented Generation (Naïve RAG) represents the foundational implementation of the RAG paradigm. It follows a basic retrieve-then-generate workflow, where documents are retrieved using traditional keyword-based methods—such as TF-IDF or BM25—from static datasets. These documents are directly passed to the language model to augment its generative output.

Naïve RAG is notable for its simplicity, making it an accessible entry point for implementing RAG systems in domains with fact-based queries and minimal contextual complexity. However, this simplicity introduces several limitations:

Lack of Contextual Awareness: Lexical matching techniques often fail to capture semantic nuances, resulting in less relevant or imprecise document retrieval.

Fragmented or Generic Outputs: Without contextual integration or document re-ranking, the generation often lacks coherence or specificity.

Scalability Challenges: Keyword-based retrieval methods are inefficient in large or diverse corpora, often returning suboptimal results.

Despite its drawbacks, Naïve RAG provides a useful baseline for evaluating more advanced RAG architectures that incorporate semantic retrieval, multi-stage reasoning, or adaptive feedback loops

2-Advanced RAG systems improve upon Naïve RAG by integrating **semantic retrieval** and **context-aware generation**. They use dense retrievers like **DPR** and neural re-rankers for better document relevance.

Main Features:

- **Dense Vector Search**: Represents queries and documents in vector space for semantic matching.
- **Contextual Re-Ranking**: Neural models re-order retrieved results based on relevance.
- **Iterative Retrieval**: Enables reasoning over multiple documents in multi-hop setups.

These systems are well-suited for **complex tasks** like research and personalized recommendations. However, they still face issues such as **high computational cost** and **scalability limits**.

3-Modular RAG is a flexible, customizable evolution of RAG systems. It breaks down the pipeline into **independent components**, allowing for **domain-specific optimization** and **tool integration**

Key Features:

- **Hybrid Retrieval**: Combines sparse (e.g., BM25) and dense (e.g., DPR) retrieval for broader query coverage.
- **Tool Integration**: Supports external APIs or tools for tasks like real-time data access or computation.
- **Composable Pipelines**: Components (retrievers, generators, re-rankers) can be easily swapped or reconfigured.

Example: A financial Modular RAG system may fetch livestock data via APIs, analyze trends with dense retrieval, and generate insights using a domain-specific model.

This architecture offers **scalability, adaptability**, and **high performance** for complex, multi-domain applications.

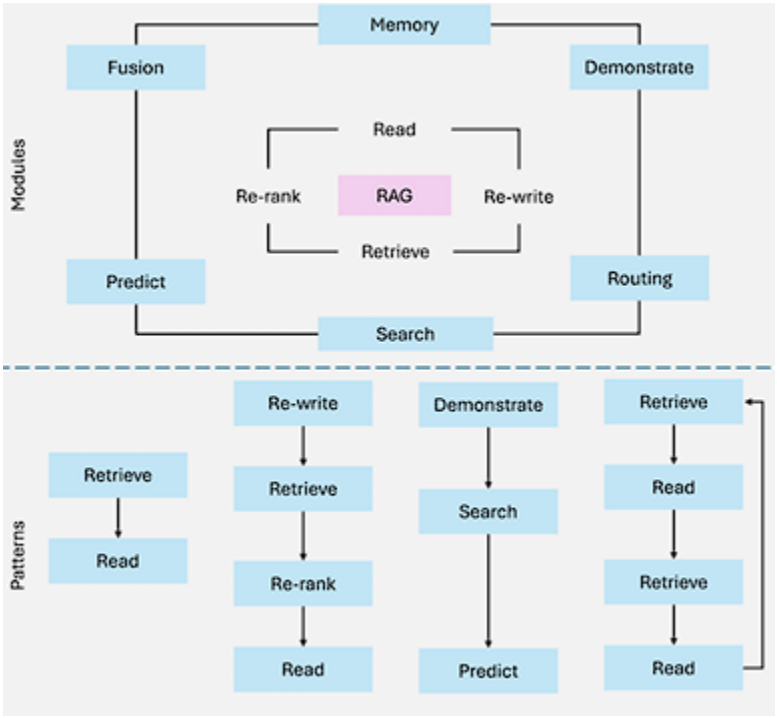


Fig. 0-3. modular Rag architecture

4-Agentic RAG introduces **autonomous agents** capable of **dynamic decision-making** and **adaptive retrieval**, enabling it to handle complex, real-time, and multi-domain queries. It builds on modular RAG by adding agent-based control and iterative feedback.

Key Features:

- **Autonomous Retrieval**: Agents choose strategies based on query complexity.
- **Iterative Refinement**: Feedback loops enhance accuracy and relevance.
- **Workflow Optimization**: Tasks are dynamically managed for efficiency.

Challenges:

- **Coordination Complexity**: Requires advanced orchestration between agents.
- **High Overhead**: Resource-intensive, especially with multiple agents.
- **Scalability Strain**: Dynamic behaviors can impact performance at scale.

Agentic RAG is ideal for applications needing high adaptability, such as **customer service**, **financial analytics**, and **intelligent tutoring systems**.

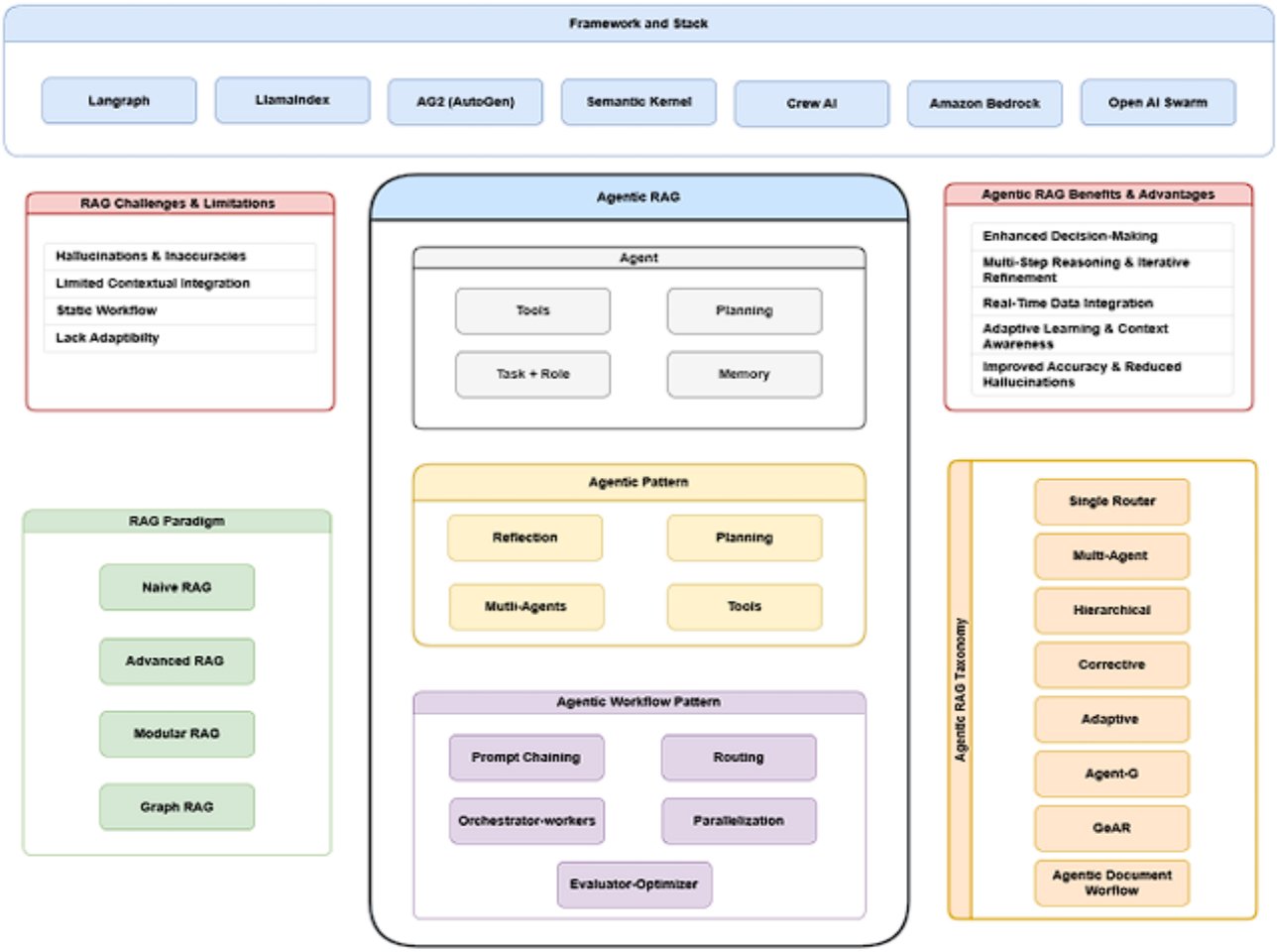


Fig. 0-4. Agentic RAG

5.4.2 Agent

Agentic Intelligence forms the foundation of Agentic Retrieval-Augmented Generation (RAG) systems, enabling them to transcend the static and reactive nature of traditional RAG. By integrating autonomous agents capable of dynamic decision-making, iterative reasoning, and collaborative workflows, Agentic RAG systems exhibit enhanced adaptability and precision. This section explores the core principles underpinning agentic intelligence.

Components of an AI Agent. In essence, an AI agent comprises

• LLM (with defined Role and Task): Serves as the agent's primary reasoning engine and dialogue interface. It interprets user queries, generates responses, and maintains coherence.

• Memory (Short-Term and Long-Term): Captures context and relevant data across interactions. Short-term memory tracks immediate conversation state, while long-term memory stores accumulated knowledge and agent experiences.

• Planning (Reflection & Self-Critique): Guides the agent's iterative reasoning process through reflection, query routing, or self-critique, ensuring that complex tasks are broken down effectively.

• Tools Vector Search, Web Search, APIs, etc.): Expands the agent's capabilities beyond text generation, enabling access to external resources, real-time data, or specialized computations

5.4.3 Agentic Pattern

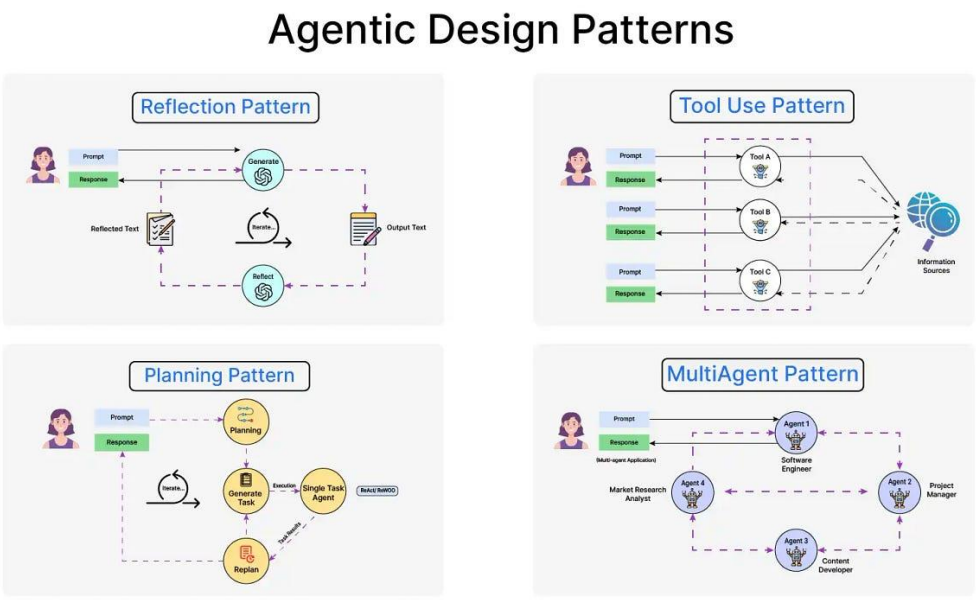


Fig. 0-5.agent design pattern

Agentic Patterns provide structured methodologies that guide the behavior of agents in Agentic Retrieval Augmented Generation (RAG) systems. These patterns enable agents to dynamically adapt, plan, and collaborate, ensuring that the system can handle complex, real-world tasks with precision and scalability. Four key patterns underpin agentic workflows:

1- Reflection

Reflection is a core agentic pattern that enables agents to iteratively evaluate and improve their output through self-feedback. By assessing their own responses for correctness, clarity, and efficiency, agents can identify and address errors or inconsistencies across tasks such as code generation, text writing, and question answering.

In practice, reflection involves prompting an agent to criticize its own output and revise it based on the feedback. This process can be enhanced by incorporating external validation tools, such as unit tests or web search, to detect gaps and confirm accuracy.

In multi-agent setups, reflection may involve distinct roles—one agent generates, while another evaluates—enabling collaborative refinement. This approach has shown measurable performance improvements in recent research, including Self-Refine, Reflexion, and CRITIC.

2- Planning

Planning enables agents to break down complex tasks into smaller steps and determine their execution order dynamically. This supports multi-hop reasoning and flexible problem-solving in uncertain contexts.

While it offers adaptability, planning can lead to unpredictable outcomes compared to structured patterns like reflection. It is best suited for tasks that require real-time adjustment rather than fixed workflows.

3- Tool Use

Tool use allows agents to interact with external APIs, tools, or data sources to go beyond their pre-trained knowledge. This enables them to perform tasks like data retrieval, computation, and system integration with greater accuracy and relevance.

Modern agents can autonomously select and use tools, supported by advancements like function calling. However, choosing the right tool remains a challenge, especially in tool-rich environments—an issue addressed by heuristic-based methods inspired by RAG.

4- Multi-Agent

Multi-agent collaboration is a key design pattern in agentic workflows that enables task specialization and parallel processing.

Agents communicate and share intermediate results, ensuring the overall workflow remains efficient and coherent. By distributing subtasks among specialized agents, this pattern improves the scalability and adaptability.

Multi-agent systems allow developers to decompose intricate tasks into smaller, manageable subtasks assigned to different agents. This approach not only enhances task performance but also provides a robust framework for managing complex interactions. Each agent operates with its own memory and workflow, which can include the use of tools,

reflection, or planning, enabling dynamic and collaborative problem-solving. While multi-agent collaboration offers significant potential, it is a less predictable design pattern compared to more mature workflows like Reflection and Tool Use.

Nevertheless, emerging frameworks such as AutoGen, Crew AI, and LangGraph are providing new avenues for implementing effective multi-agent solutions.

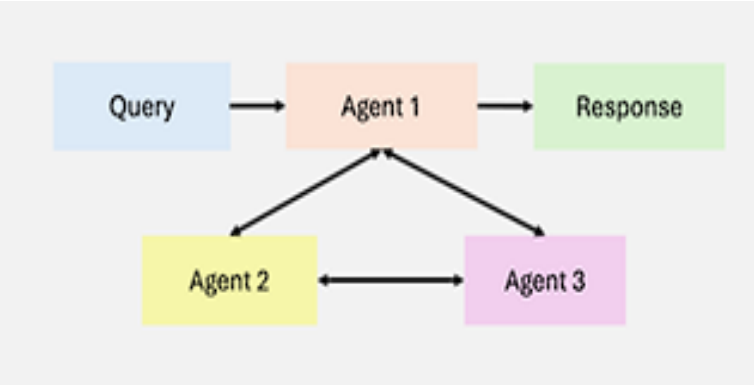


Fig. 0-6.overview of multiagent

These design patterns form the foundation for the success of Agentic RAG systems. By structuring workflows—from simple, sequential steps to more adaptive, collaborative processes—these patterns enable systems to dynamically adapt their retrieval and generative strategies to the diverse and ever-changing demands of real-world environments. Leveraging these patterns, agents are capable of handling iterative, context-aware tasks that significantly exceed the capabilities of traditional RAG systems.

5.4.4 Agentic Workflow Pattern

Agentic workflow patterns: structure LLM-based applications to optimize performance, accuracy, and efficiency. Different approaches are suitable depending on task complexity and processing requirements.

Prompt Chaining Enhancing Accuracy Through Sequential Processing Prompt chaining decomposes a complex task into multiple steps, where each step builds upon the previous one.

This structured approach improves accuracy by simplifying each subtask before moving forward. However, it may increase latency due to sequential processing

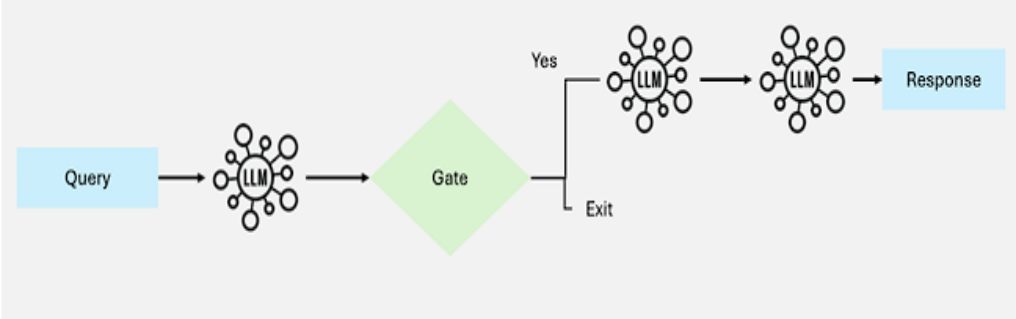


fig. 0-7.prompt chaining overflow

Routing: Directing Inputs to Specialized Processes

Routing involves classifying an input and directing it to an appropriate specialized prompt or process. This method ensures distinct queries or tasks are handled separately, improving efficiency and response quality.

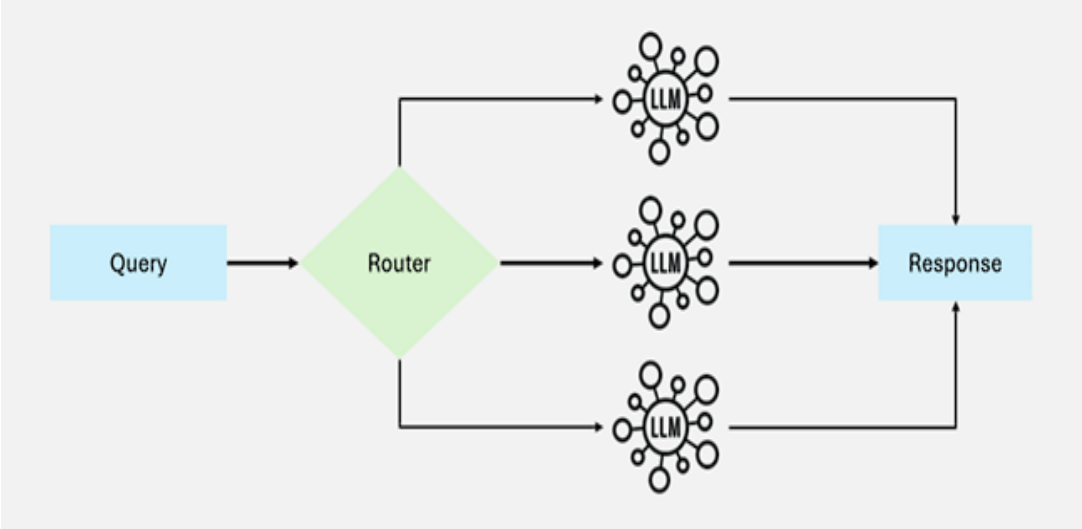


Fig. 0-8. routing flow

Parallelization: Speeding Up Processing Through Concurrent Execution

Parallelization divides a task into independent processes that run simultaneously, reducing latency and improving throughput. It can be categorized into sectioning (independent subtasks) and voting (multiple outputs for accuracy)

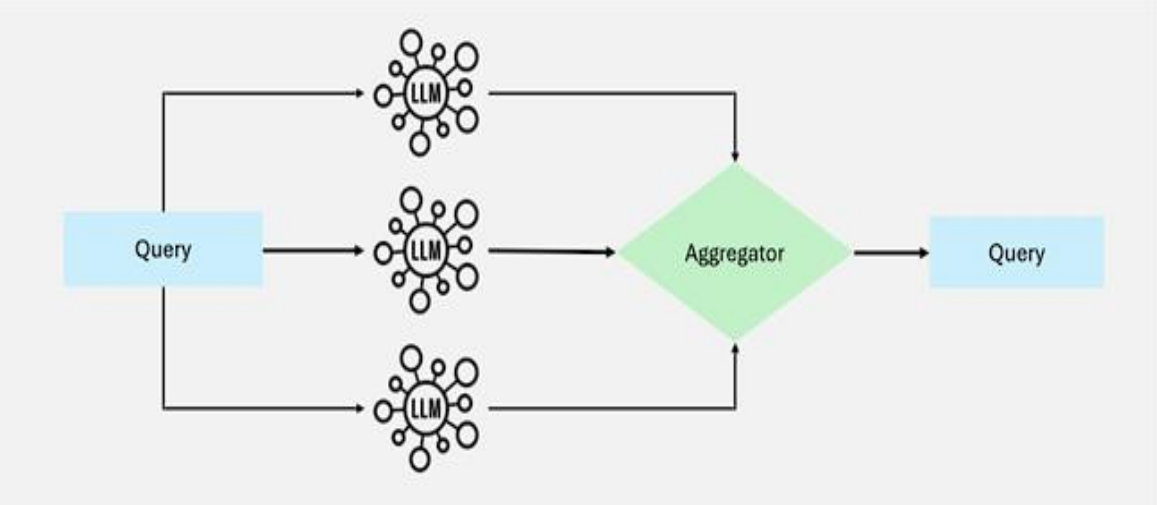


Fig. 0-9. palleralzation workflow

Orchestrator-Workers: Dynamic Task Delegation

This workflow features a central orchestrator model that dynamically breaks tasks into subtasks, assigns them to specialized worker models, and compiles the results. Unlike parallelization, it adapts to varying input complexity.

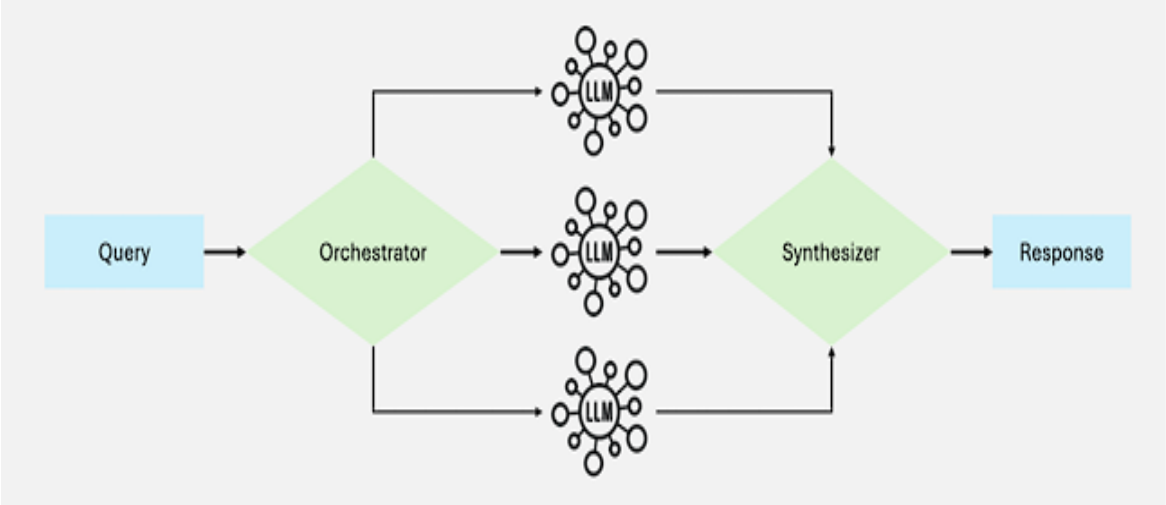


Fig. 0-10. Orchestrator-Workers workflow

Evaluator-Optimizer: Refining Output Through Iteration

The evaluator-optimizer [12, 13] workflow iteratively improves content by generating an initial output and refining it based on feedback from an evaluation model

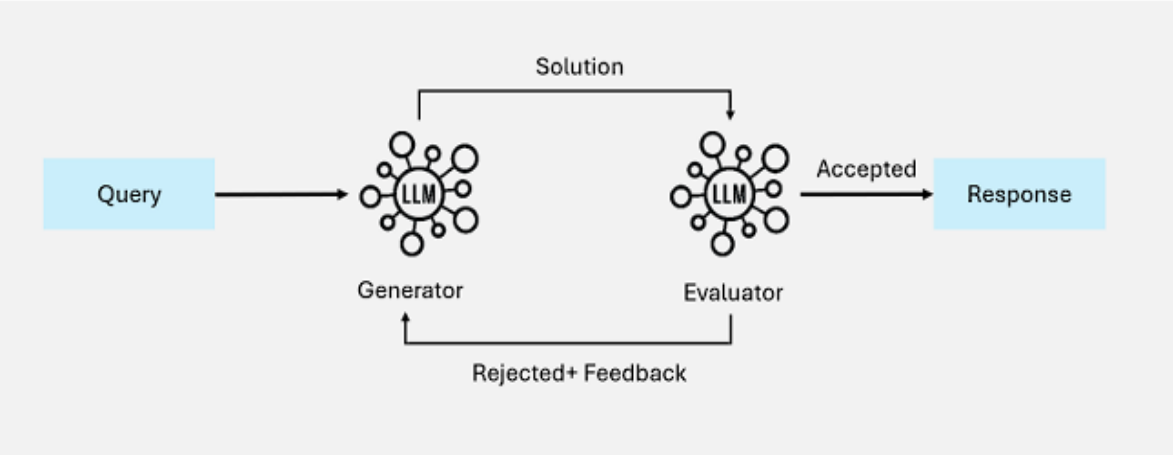


Fig. 0-11. evaluator optimize workflow

5.5 AI Framework

A **framework** is a system that enhances the capabilities of **Large Language Models (LLMs)** by enabling them to access and incorporate information from external data sources. These frameworks are essential for building intelligent, real-world applications where LLMs need to interact with documents, APIs, or databases to provide accurate, context-aware responses.

During the development of our system, we explored several widely used frameworks for LLM-based application development, including **Lang Chain**, **Haystack**, and **Llama Index**. After evaluating their features and compatibility with our project goals, we chose **Llama Index** as the core framework for implementing **context-augmented LLM functionality**.

How Llama Index Works

Llama Index provides an end-to-end workflow that allows developers to connect LLMs with external data. Its architecture is modular, making it easy to integrate with different data sources and LLM providers. Below is a breakdown of its core components:

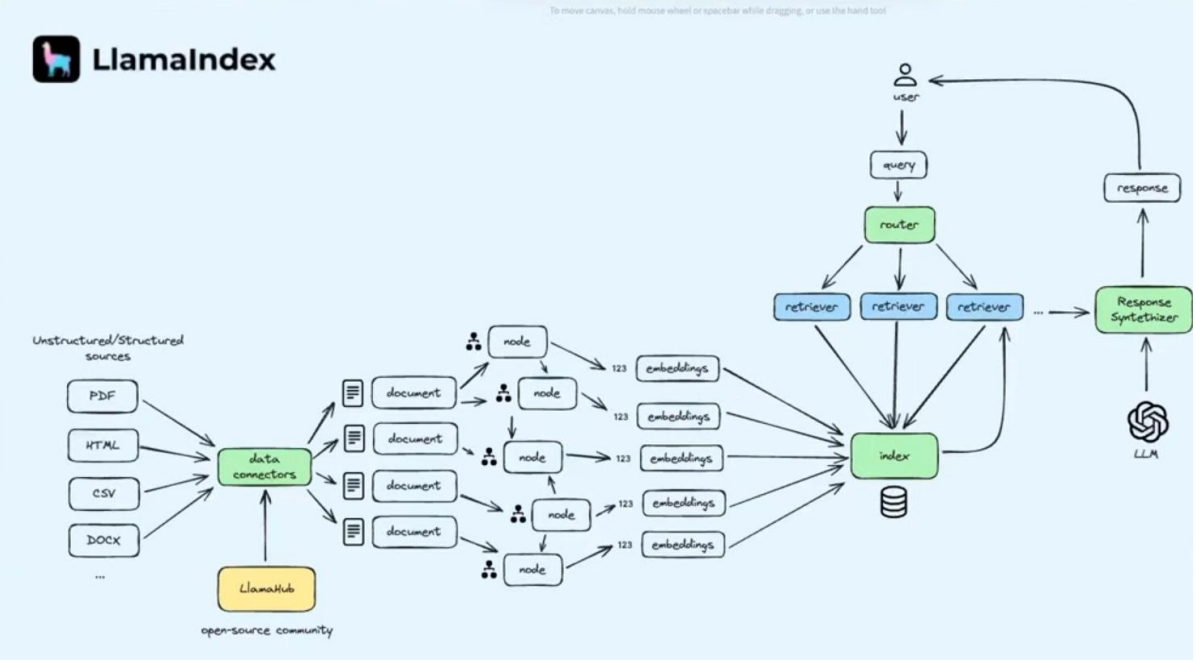


Fig. 0-12. how llama index works

- **1. Loading:**
  Ingest data from various sources such as PDFs, APIs, or databases using built-in connectors.
- **2. Parsing:**
  Split documents into smaller, structured units (called **nodes**) using parsers and text splitters.
- **3. Indexing:**
  Convert nodes into vector embeddings or graph structures and store them in specialized indices for efficient retrieval.

- **4. Querying:**
  Use query engines or chat interfaces to retrieve relevant nodes and pass them to LLMs for response generation.
- **5. Storing:**
  Persist embeddings, documents, and index metadata to support reuse and scalability across sessions.
- **6. Advanced Components:**
  Support for agents, multi-step workflows, evaluation tools, and observability for monitoring and debugging.
- **7. Deployment:**
  Deploy applications using **Llama Deploy**, enabling production-grade performance and scalability.

retrieval.
- **Indexing**: Nodes are embedded and stored in vector or graph-based indices for efficient
- **Advanced Components**: Includes support for agents, workflows, evaluation tools, and observability.

Summary

Llama Index provides a robust, flexible pipeline for building LLM-based applications. It supports both **simple prototypes** and **complex, scalable systems**, making it ideal for projects that require reliable context retrieval and seamless integration with LLMs. Its modular architecture and extensive tooling enable developers to build intelligent assistants, RAG pipelines, and interactive AI applications efficiently.

5.6 Data Indexing (Vector Database)

To enable efficient and accurate retrieval of information from large collections of unstructured data, our system employs **vector-based indexing** using a **vector database**. This process transforms raw text into numerical representations (embeddings) that capture semantic meaning, enabling similarity-based search rather than relying on traditional keyword matching.

5.6.1 How It Works:

1. **Text Embedding:**
   Each document (or document chunk) is processed by an embedding model, such as sentence-transformers, which converts the text into a high-dimensional vector. These vectors capture the **contextual meaning** of the content.
2. **Index Creation:**
   The generated vectors are stored in a **vector database** (e.g., FAISS, Chroma, or other supported backends via LlamaIndex). These indices allow for **fast and efficient similarity search** using algorithms like approximate nearest neighbor (ANN).
3. **Contextual Retrieval:**
   When a user submits a query, it is also embedded into a vector. The system then

compares this query vector with the stored document vectors to find the most **semantically similar** content. This enables the LLM to generate context-aware, relevant responses.

5.6.2 Why Vector Indexing?

- **Scalability:** Can handle large amounts of data efficiently
- **Speed:** Rapid similarity searches even in massive datasets
- **Accuracy:** Returns results based on meaning, not just keywords
- **Flexibility:** Works across different data formats and domains

5.6.3 Integration in Our System:

In our RAG-based system, vector indexing is a **core component**. It allows the LLM to access and reason over complex telecommunications content (e.g., Open RAN, MIMO standards) with **precision and speed**. This significantly enhances the model's ability to provide **accurate**, **domain-specific**, and **contextual answers**.

5.7 Our Project: Content Optimization RAG System

5.7.1 **Overview**

Our graduation project is a practical implementation of a **Retrieval-Augmented Generation (RAG)** system, specifically designed for **content optimization** in highly technical domains. The core idea of this system is to **bridge the gap** between **unstructured technical data** and **human-understandable insights** by combining the power of modern **Large Language Models (LLMs)** with advanced **semantic retrieval** techniques.

In traditional setups, general-purpose LLMs often generate hallucinated or non-specific outputs when dealing with complex, niche topics such as Open RAN or MIMO. Our system addresses this challenge by **augmenting the LLM with accurate, domain-grounded context**, retrieved in real-time from structured and unstructured sources such as PDF guidelines and web documents.

5.7.2 Problem Statement

In fields such as telecommunications, technical documentation is often dense, highly specialized, and difficult to navigate. Engineers and researchers may need to spend significant time reading and interpreting large documents. Moreover, general LLMs may fail

to understand this content accurately due to the lack of training on specialized terminology or specifications.

Our goal was to build a system that can:

- Interpret complex technical documents
- Answer queries using up-to-date, relevant context
- Generate clear, structured reports summarizing key insights
- Support local or API-based deployment while preserving privacy

5.7.3 System Architecture

The system follows a **modular architecture**, which allows flexibility, scalability, and clarity of implementation. It is divided into several layers:

1. **Data Layer**:
    - PDF guidelines and web content are collected and stored in a data/pdfs/ directory.
    - The data_loader.py script reads and prepares this content for further processing.
2. **Indexing Layer**:
    - Documents are split into smaller chunks or "nodes" using LlamaIndex's parser and text splitter.
    - These nodes are embedded into high-dimensional vectors using language embeddings.
    - The embeddings are stored in **Chroma**, a fast vector database, through the chroma_integration.py and index_builder.py modules.
3. **Retrieval and Generation Layer**:
    - When a user submits a prompt, it is converted into a vector and matched against stored document vectors.
    - The most relevant nodes are retrieved and passed to an LLM (e.g., LLaMA 3.2 via OpenRouter).
    - The query_engine.py handles this process, coordinating between retrieval and generation steps.
4. **Output Layer**:
    - The system generates a detailed **Markdown report** (500–600 words), along with a summary section (100–150 words).
    - These reports are automatically converted into **PDF files** using **Pandoc** and **LaTeX**.
5. **User Interface**:
    - Users can interact with the system either via a **command-line interface (CLI)** (main.py) or a **web-based interface** (index.html served through main_api.py).

- This allows the system to be accessible and easy to use even by non-technical users

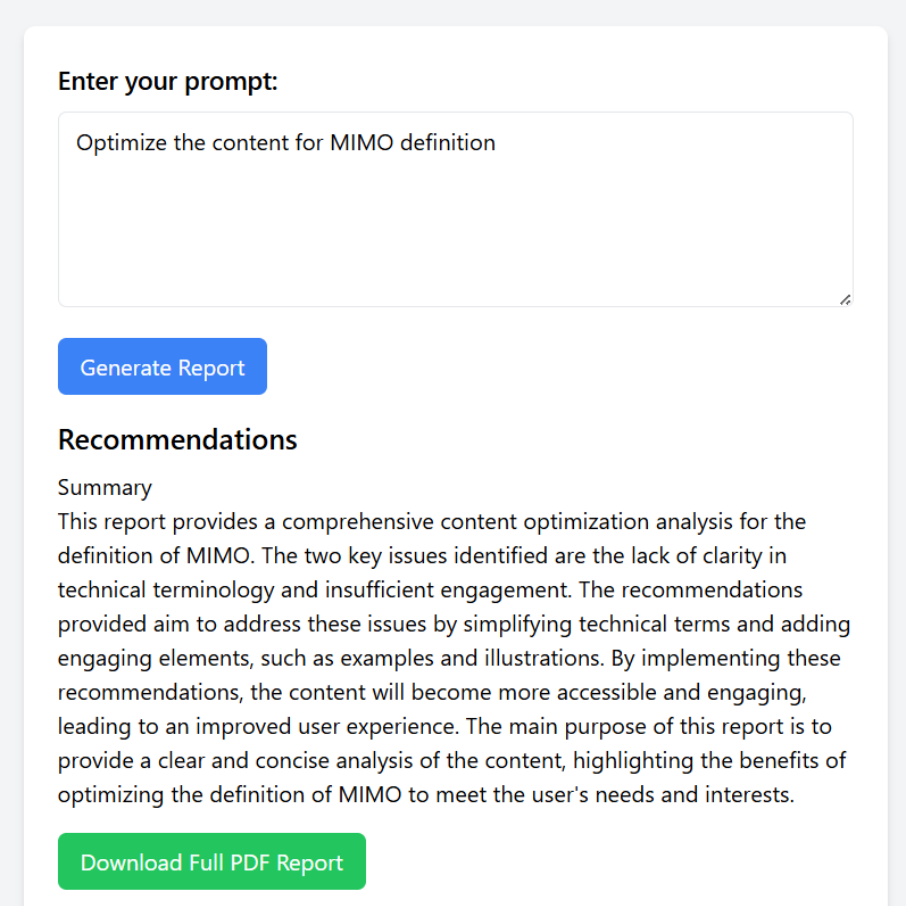


Fig. 0-13. content optimization tool

It plays a central role in the usability and accessibility of the system. Its importance includes:

- **User Accessibility**: Enables non-technical users to interact with powerful language models without using command-line tools or APIs.

- **Real-Time Interaction**: Provides a live interface for entering queries, viewing results, and engaging in multi-turn conversations.

- **Transparency**: Allows users to see how responses are generated, including relevant retrieved documents or references.

- **Debugging and Evaluation**: Offers developers an easier way to test model behavior, identify issues, and visualize model outputs.

- **Productization**: Makes the system more deployable as a complete application rather than a backend service.

5.7.4 Key Features

- **Context-Aware Generation**: LLM responses are generated based on relevant document chunks, reducing hallucinations.
- **Markdown-to-PDF Reporting**: Ensures professionally formatted outputs suitable for academic or industrial use.
- **Semantic Search via Vector Embeddings**: Delivers more accurate and meaning-based document retrieval.
- **Web Interface**: Simple UI for non-developers to interact with the system without coding.
- **Modular Backend**: Each functionality (loading, indexing, querying, generating) is isolated in separate Python modules.

5.7.5 Use Case Example

A typical use case would involve a user inputting a prompt such as: **"Analyze the clarity and structure of the NONRTRIC components page from the O-RAN SC documentation."**

The system then:

1. Retrieves matching content from PDF and web sources
2. Passes it to the LLM
3. Generates a report with insights and recommendations
4. Provides a downloadable PDF file of the report

Strengths and Contributions

- **Accuracy**: By grounding answers in actual documents, we minimize false or made-up content.
- **Efficiency**: Saves time by automating the process of reading and summarizing long technical content.
- **Adaptability**: System can be reused in other domains by changing the document sources.
- **Scalability**: With Chroma and LlamaIndex, the system supports large datasets and complex queries.

5.7.6 Summary

Our RAG-based content optimization system showcases the real-world potential of LLMs when combined with retrieval technologies. It allows professionals to extract valuable insights from complex datasets through natural language interaction, while maintaining accuracy, clarity, and usability. The system is not just a prototype but a fully functional, extensible framework ready for further development or real-world deployment.

5.3 future work

- Building upon the current implementation of energy-saving use cases within the ns-3 and 5G Lena simulation environment, several future directions can be explored to
- extend the project's impact and practical applicability:
- 1-**Integration with Real-Time and Non-RT RIC:** Extend the current simulation-based control of antenna port power into a real O-RAN testbed using Non-RT and Near-RT RIC to dynamically manage energy savings based on live network conditions and traffic variations.
- 2-**Agentic AI Advanced Policy Development:** Develop AI-driven optimization policies to decide dynamically which antenna ports to disable during low-traffic periods while maintaining acceptable QoS metrics. This includes reinforcement learning models to adapt power configurations based on observed network behavior and traffic load patterns.
- 3-**Multi-Objective Optimization:** Expand the current energy-saving approach to include additional objectives, such as reducing interference, improving user experience, and managing thermal constraints within base stations, using advanced multi-objective AI algorithms.
- 4-**Fine-Grained Power Control:** While the current implementation supports port-level power control, future work can target fine-grained power distribution across sub-carriers or individual resource blocks, using enhanced PHY layer control mechanisms.
- 5-**End-to-End Cloud-Based Orchestration:** Leverage cloud-native O-RAN environments to orchestrate energy-saving use cases, automating deployment, monitoring, and adjustment using Kubernetes, Helm, and Istio, aligning with production-grade O-RAN ecosystems.
- 6-**Testing with Advanced Traffic Models:** Validate the energy-saving scenarios under different realistic traffic patterns (e.g., bursty IoT traffic, video streaming) to evaluate robustness and further optimize energy vs. performance trade-offs.
- 7-**Security Considerations:** Investigate security implications of dynamically enabling/disabling antenna ports within O-RAN to prevent potential attack surfaces, ensuring that energy-saving use cases do not compromise network integrity

References

1. https://cttc-lena.gitlab.io/nr/cttc-nr-demo-tutorial.pdf (NR tutorial)
2. https://drive.google.com/file/d/1LN9Gz3yyN-eRHAtb3_FNalAjHqXqRisV/view?usp=sharing
3. https://www.techplayon.com/5g-nr-antenna-port-logical-and-physical-antenna-mapping/
4. https://arxiv.org/pdf/2404.17472
5. https://docs.o-ran-sc.org/projects/o-ran-sc-nonrtric-plt-ranpm/en/latest/datafilecollector/index.html/
6. https://docs.o-ran-sc.org/projects/o-ran-sc-nonrtric-plt-ranpm/en/latest/pm-file-converter/index.html
7. https://docs.o-ran-sc.org/projects/o-ran-sc-nonrtric-plt-ranpm/en/latest/pmproducer/index.html
8. https://docs.o-ran-sc.org/projects/o-ran-sc-nonrtric-plt-ranpm/en/latest/influxlogger/index.html
9. https://docs.o-ran-sc.org/projects/o-ran-sc-nonrtric-plt-informationcoordinatorservice/en/latest/index.html
10. https://arxiv.org/abs/2407.09619
11. https://ieeexplore.ieee.org/document/10646355
12. https://openrangym.com/
13. https://github.com/tkn-tub/ns3-gym
14. https://ieeexplore.ieee.org/document/9700651
15. https://groups.google.com/g/5g-lena-users
16. https://o-ran-sc.org/
17. https://docs.o-ran-sc.org/en/e-release/projects.html

[18] A. Singh, A. Ehtesham, S. Kumar, and A. Talaei-Khoei, “Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG,” arXiv preprint arXiv:2501.09136, Jan. 2024. [Online]. Available: https://arxiv.org/abs/2501.09136
[19] LlamaIndex Documentation, “Connecting LLMs with external data.” [Online]. Available: https://www.llamaindex.ai
[20] Ollama Documentation, “Local deployment of open-source LLMs.” [Online]. Available: https://ollama.com
[21] P. Lewis, B. Oguz, S. Riedel, et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” arXiv preprint arXiv:2005.11401, May 2020. [Online]. Available: https://arxiv.org/abs/2005.11401

Thanks