**Research Project final Presentation**

**Slide 4:**

A diagram of a computer program

Description automatically generated with medium confidence

An Open Radio Access Network (O-RAN) is a disaggregated approach to deploying mobile fronthaul and midhaul networks built entirely on cloud native principles. O-RAN underscores streamlined 5G RAN performance objectives through the common attributes of efficiency, intelligence and versatility. Open RAN deployed at the network edge will benefit 5G applications such as autonomous vehicles and the IoT, support network slicing use cases effectively, and enable secure and efficient over-the-air firmware upgrades.

O-RAN is an evolution of the Next Generation RAN (NG-RAN) architecture, first introduced by the GSMA’s 3GPP in their release 15 (5G version 1) technical specification TS 38.401. The O-RAN Alliance formed to undertake the advancement of NG-RAN philosophies, expanding on the scope of what was originally outlined by the 3GPP. Comprising over 1601 member companies, the O-RAN alliance issues specifications and releases open source software under the auspices of the Linux Foundation.

**Slide 5:**

Logo, circle

Description automatically generated

O-RAN ALLIANCE is a world-wide community of mobile operators, vendors, and research & academic institutions operating in the Radio Access Network industry. As the RAN is an essential part of any mobile network, O-RAN ALLIANCE's mission is to re-shape the industry towards more intelligent, open, virtualized and fully interoperable mobile networks. The new O-RAN standards enable a more competitive and vibrant RAN supplier ecosystem with faster innovation to improve user experience. O-RAN-compliant mobile networks will at the same time improve the efficiency of RAN deployments as well as operations by the mobile operators. To achieve this, O-RAN ALLIANCE publishes new RAN specifications, releases open software for the RAN, and supports its members in integration and testing of their implementations. For more information please visit www.o-ran.org.

Link: <https://www.o-ran.org/>

**Slide 7:**

Diagram

Description automatically generated

**Service Management and Orchestration (SMO):**

SMO must ensure the Non-RT RIC to access specific functionalities related to RAN optimization actions like in particular collecting Performance Measurements (PM) through O1 and O2 interfaces.

In addition to this the SMO must take care of the orchestration of the Network Functions Virtualization Infrastructure (NFVI), managing the life cycle of O-RAN network elements (Near-RT RIC, O-CU, O-DU, O-RU) which can be either Virtual Network Functions (VNFs) hosted in specific location of the O-Cloud infrastructure or Physical Network Functions (PNFs) exposed by cell sites.

For non-virtualized parts, typically O-RU functionalities which are related to area coverage and need to be placed at cell sites, the SMO supports the deployment of physical network elements on dedicated physical resources with management through the O1 interface.

For virtualized network elements, the SMO has the capability to interact with the O-Cloud to perform network element life cycle management, for

example it can instantiate the virtualized network element on the target infrastructure through the O2 interface or indicate the selected geo-location for each VNF to be instantiated.

Finally the Service Management and Orchestration framework must be able

to support the communication between the deployed network elements and so

it is in charge of IP addressing, network reconfiguration and system updates.

Then to guarantee various deployment solutions the Operation and Maintenance architecture defined by O-RAN describes in details the requirements

needed such that the SMO framework can be provided by third-party Network

Management Systems (NMS) or orchestration platforms like for example the

Linux Foundation’s Open Network Automation Platform (ONAP) [20].

**near-RT RIC:** O-RAN near-real-time RAN Intelligent Controller: a logical function that enables near-real-time control and optimization of O-RAN elements and resources via fine-grained data collection and actions over E2 interface.

**non-RT RIC:** O-RAN non-real-time RAN Intelligent Controller: a logical function that enables non-real-time control and optimization of RAN elements and resources, AI/ML workflow including model training and updates, and policy-based guidance of applications/features in near-RT RIC.

**O-CU:** O-RAN Central Unit: a logical node hosting RRC, SDAP and PDCP protocols

RRC= Radio Resource Control

SDAP= Service Data Adaptation Protocol

PDCP= Packet Data Convergence Protocol

**O-CU-CP:** O-RAN Central Unit – Control Plane: a logical node hosting the RRC and the control plane part of the PDCP protocol.

**O-CU-UP:** O-RAN Central Unit – User Plane: a logical node hosting the user plane part of the PDCP protocol and the SDAP protocol.

**O-DU:** O-RAN Distributed Unit: a logical node hosting RLC/MAC/High-PHY layers based on a lower layer functional split.

**O-RU:** O-RAN Radio Unit: a logical node hosting Low-PHY layer and RF processing based on a lower layer functional split. This is similar to 3GPP’s “TRP” or “RRH” but more specific in including the Low-PHY layer (FFT/iFFT, PRACH extraction).

**O1:** Interface between management entities in Service Management and Orchestration Framework and O-RAN managed elements, for operation and management, by which FCAPS management, Software management, File management shall be achieved.

**O1\*:** Interface between Service Management and Orchestration Framework and Infrastructure Management Framework supporting O-RAN virtual network functions.

**Slide 8:**

Timeline

Description automatically generated

**Near-Real-time RIC X-APPs (RICAPP):** Expand the community working on open source xApps for O-RAN SC. Enhance the set of open source xApps in support of the R-SAC use cases (traffic steering, network slicing) as well new use cases. Update and enhance existing xApps to take advantage of the new features in xApp SDK (implemented by the xApp frameworks in C++, go, and python).

**Non-Real-time RIC (A1 & R1 Interfaces) (NONRTRIC):**

* The primary goal of Non-RT RIC is to support intelligent RAN optimization by providing policy-based guidance, ML model management and enrichment information to the near-RT RIC function so that the RAN can optimize, e.g., RRM under certain conditions.
* It can also perform intelligent radio resource management function in non-real-time interval (i.e., greater than 1 second).
* Non-RT RIC can use data analytics and AI/ML training/inference to determine the RAN optimization actions for which it can leverage SMO services such as data collection and provisioning services of the O-RAN nodes.
* Non-RT-RIC will define and coordinate rApps (Non-RT-RIC applications) to perform Non-RT-RIC tasks.
* Non-RT-RIC will host the A1 interface (between NONRTRIC & near-RT RICs )
* Non-RT-RIC will also host the new R1 interface (between rApps and SMO/NONRTRIC services)

**Slide 9:**

F Release:

The F release is completed and its source code is maintained within the master branch of each repo.

• Near-Real-time RIC X-APPs (RICAPP)

• Near-Real-time RAN Intelligent Controller Platform (E2 Interface) (RICPLT)

• Non-Real-time RIC (A1 & R1 Interfaces) (NONRTRIC)

• Operation and Maintenance (OAM)

• O-RAN Central Unit (OCU)

• O-DU High

• O-DU Low

• Simulators (SIM)

• Infrastructure (INF)

• Integration and Test (INT)

• Documentation (DOC)

• Service Management and Orchestration (SMO)

**Slide 8:**

Kubernetes: Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management.[3][4] Originally designed by Google, the project is now maintained by the Cloud Native Computing Foundation.

Helm: Helm is a package manager for Kubernetes. Helm helps you manage Kubernetes applications — Helm Charts help you define, install, and upgrade even the most complex Kubernetes application.

**Slide 9:**

Here,all necessary service pods are running. Importantly, a1 mediator,

armanager, appmgr and 01 mediator are running. Which are responsible

for building connections with non-RT RIC.

**Slide 10:**

As like as Slide 8

**Slide 11:**

Now we can see that all pods are in the running stage. Where

the A1 simulator, controller, helmanager and other necessary Kubernetes

pods are active. It means the non-RT RIC platform is ready to use.

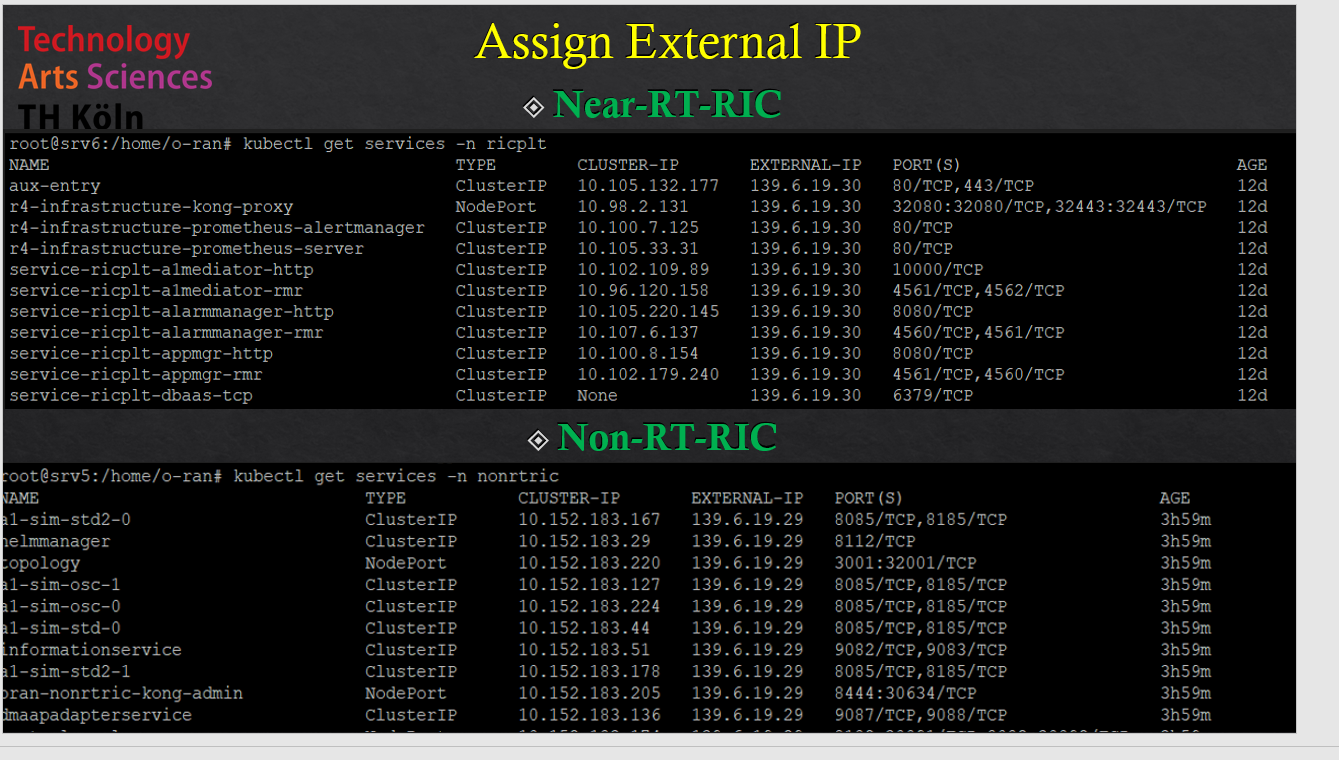
**Slide 12: Connection set-up**

To build the connection between near-RT RIC and non-RT RIC we need to follow a few steps. Which is guided by the software community of o-ran Alliance.

* Assign external IP to near and non RT RIC Services.
* Edit *application\_configuration.json* file (non-rt-ric).
* ONAP Components Installation.
* Replace *policy\_schema\_ratecontrol.json* file with Policy Control file.

**Slide 13:**

To connect a near-RT RIC with a non-RT RIC platform, it is mandatory to assign an external IP address to all components of both RIC.



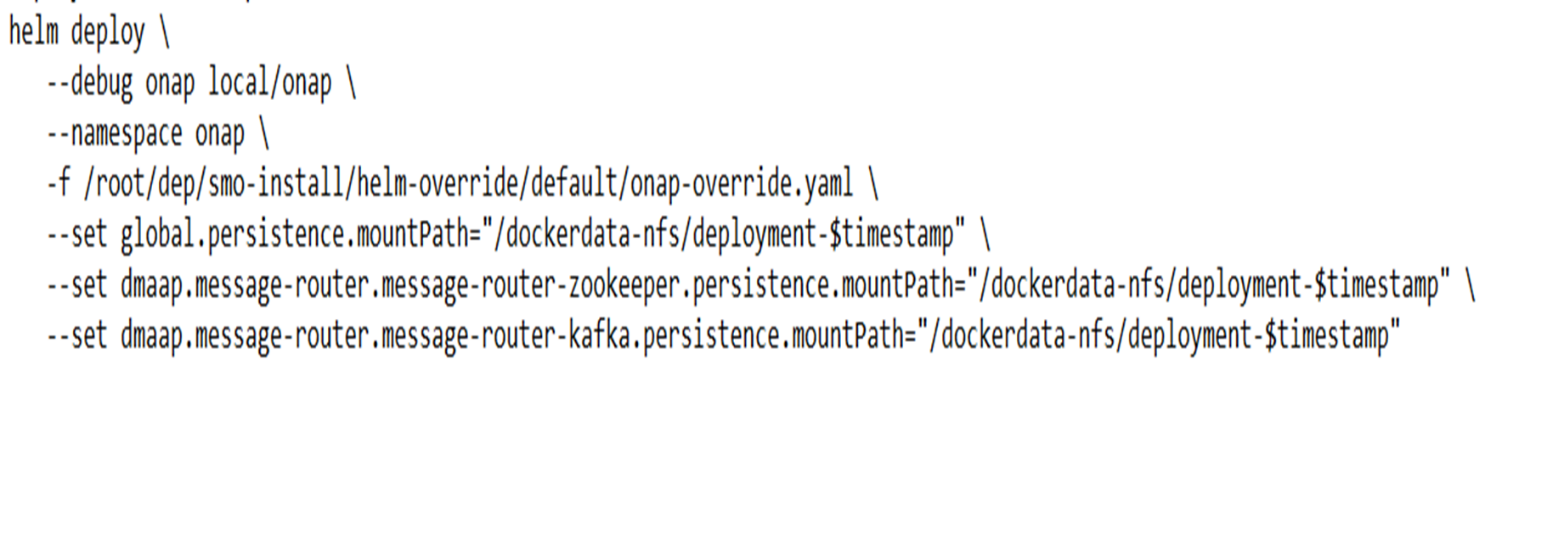
**Slide 14:**

A screenshot of a computer

Description automatically generated

**Slide 15: ONAP Components Installation**

To create a policy, we need to update the policy schema file. This file is provided by default. Apart from that, ONAP has developed a policycreating platform. So, before we create policies, we need to set up a few dependencies, which are provided by ONAP. Run the following command to deploy Helm for ONAP services.



**Slide 16: Policy Service**

Now non-rt ric is ready to create policy. First we need to edit policy schema ratecontrol.json file by replacing the following script.

Now curl this file, and once it’s done, a policy will be created, and to see the policy, we need to go to policy control on the RIC platform: For that, first run the following command to create a policy:

Now go to the non-RT RIC control panel and enter the option policy type bar. Refresh the page, and we will see that the policy has been created. In my case, the policy name is 21003.

**Slide 17: Control Panel of Non-RT RIC**

**Slide 18: Policy Configuration**

**Slide 19: Wireshark capture**

In the following Wireshark [Figure 10] capture, we will see that near-RT RIC and non-RT RIC transferring packets. Here, 139.6.19.30 is for near- RT RIC and 139.6.19.29 is for non-RT RIC platforms.

**Slide 20: Wireshark capture**

Now the [Figure 11] is about the transferred packet. Here, packet number 87806 is a transfer from non-RT RIC to near-RT RIC, which is a policy packet.

This packet was transferred through the HTTP/JSON protocol. Application/ json file presenting the data, and the policy name is 21003, which we have seen before in the RIC control panel. Here in the JavaScript Object Notation section, there is an option with the name ”array. This array contains a path and a member with the value 21003.

**Slide 21: Future Work**

Currently, the system is not prepared to connect with the 5G core network as it solely encompasses RIC components. However, the development of other components is underway, and once completed, the system will be ready for integration with the 5G core network. The O-RAN Alliance is actively involved in the development of these additional components, providing open-source implementation resources through their published

releases. As part of future work, the focus will be on deploying these forthcoming components and establishing the necessary connections between them and the 5G core network. This ongoing effort will pave the way for a comprehensive and fully functional system that leverages the capabilities of the 5G core network.

**Slide 22: Conclusion**

The objective of this project was to deploy developed components of ORAN in a specific radio access network intelligent controller part over a virtualization infrastructure to investigate the capabilities of those components as well as test their connectivity. Additionally, the study aimed to evaluate the crucial role of Kubernetes in orchestrating containerized applications. To achieve these objectives, the approach taken in this research

project primarily revolved around practical aspects, demonstrating the deployment of specific O-RAN software components in a customized virtual environment.

Despite the initial challenges, the project managed to deploy non-RT RIC, near-RT RIC, and xApp components successfully. Basic performance evaluations were conducted to assess their performance. The connectivity between the two RICs was established, allowing effective communication. Furthermore, the creation of policies was accomplished without any issues, enhancing the overall functionality of the deployed components. These achievements demonstrate significant progress in implementing the desired functionalities and capabilities of the system. The positive outcomes from the performance evaluations provide a solid foundation for further optimization and fine-tuning of the components, ensuring optimal performance in real-world scenarios.

In summary, the procedural steps taken in this research project can be seen

as a starting point for further exploration into the feasibility of deploying

O-RAN software. These steps provide a basic framework for assessing

the deployment process, which can be adapted and expanded upon as the

software components continue to evolve.